Ingeniería Tecnica Informática de Gestion

PROYECTO FIN DE CARRERA

Sistema de diálogo oral para la consulta de las Loterías y Apuestas del Estado mediante VoiceXML

Autor: Rafael Morillas García-Patos

Tutor: Dr. David Griol Barres

Leganés, junio 2015

Título: Sistema de diálogo oral para la consulta de las Loterías y Apuestas del Estado mediante VoiceXML

Autor: Rafael Morillas García-Patos

Director: Dr. David Griol Barres

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Me gustaría agradecer a todas aquellas personas que han estado a mi lado a la hora de realizar este proyecto fin de carrera, sobre todo a Naiara una persona sin la cual yo ahora mismo no tendría este proyecto fin de carrera terminado, y a mis padres Mª Isabel y Rafael dado que sin su apoyo constante no habría tenido fuerzas suficientes para continuar en los momentos de flaqueza.

Para finalizar me gustaría agradecer a mi tutor David Griol, su paciencia infinita a la hora de guiarme y enseñarme todo lo necesario para la finalización de este proyecto. Sin un tutor con esa capacidad resolutiva y sobre todo esa capacidad de animarme no creo que hubiese sido capaz de terminarlo.

A todos y cada uno de ellos Mil Gracias.

Resumen

El principal objetivo de este proyecto es acercar la información o hacer más fácil la obtención de la información a todas las personas. Ahora mismo existen muchos tipos de mecanismos para obtener información, pero cuantas más posibilidades o variantes tengamos disponibles, más fácil nos es obtenerla.

En este caso se plantea la obtención de información mediante un contestador Voz IP, es decir, se ofrece un número de teléfono desde el cual un usuario puede llamar y obtener los resultados de la lotería nacional y saber si su número ha sido premiado. Todo ello automatizado por un sistema de diálogo gestionado por un ordenador.

Este tipo de información tiene varias formas de ser consultada, por ordenador, por televisión, si eres lo bastante rápido como para seguir las letras, o bien acercándose físicamente al sitio donde compró el boleto. Hay mucha gente, que por sus condiciones físicas, no pueden usar un ordenador o les es muy complicado. Por lo tanto, este proyecto plantea una alternativa para ofrecer la información o hacer su obtención más sencilla, de esta manera todo el mundo lo tiene a su disposición siempre que quiera o siempre que tenga un móvil desde el que realizar la llamada.

Para hacer el sistema incluso más cómodo se han añadido módulos que se encargan de hacer un estudio de los usuarios que lo utilizan y ofrecer directamente los resultados sin la necesidad de realizar todo el proceso de presentación. De esta manera el sistema es más dinámico y gestiona las petición muchísimo más rápida.

La aplicación utiliza una plataforma denominada Voxeo, la cual es gestionada con un tipo de lenguaje específico llamado VoiceXML. Esta plataforma es la encarga de ofrecer la gestión de la entrada y la salida de voz de la aplicación. Además el proyecto también se apoya en otros lenguajes de programación como PHP y SQL. PHP es el encargado de llevar la gestión de las elecciones y SQL es el que nos permite la comunicación con la base de datos instalada en el sistema.

Tras el análisis y el estudio de los diferentes lenguajes utilizados se ha desarrollado la aplicación y se ha instalado en un servidor web. Una vez completado esto se ha realizado un estudio de viabilidad del sistema con la ayuda de algunos usuarios que han testeado la aplicación.

Palabras clave: VoiceXML, Sistemas de diálogo, Interacción oral, Voxeo, Loterías y apuestas, Información.

Abstract

The main objective of this project is to bring the information closer or make the information more obtainable for everyone. At the moment a lot of ways of obtaining information exist, but if we offer more possibilities it could be easier to obtain this kind.

In this project I aim to show another kind of method of getting information. For example, a voice over IP answering machine, that is, you can call a telephone number and the answering machine answers you and gives you the result of the national lottery. Furthermore the answering machine is able to know if your lottery numbers have won the prize. The machine is automatic and it is managed by a computer.

The information has some methods of being obtained, like computer or a television (if you read faster than a television shows you), or you can go wherever you can to buy a ticket. A lot of people can´t use some of these methods, maybe as a result of their physical problems or maybe because they don´t know how to use a computer very well. So this project puts forward another method of obtaining information in the easiest possible way. Everyone who has a mobile phone is able to obtain the information.

To make the system more comfortable, the project has some modules which are responsible for studying the user who use the system. These modules recognize the lottery that the users play every time and it shows the result directly without having to go through the presentation process.

The application uses a platform called Voxeo, which is managed with a specific type of programming language called VoiceXML. This platform is responsible for providing management of input and output voice sounds. In addition the project also uses other programming languages like PHP and SQL. PHP brings the way of the execution and SQL allows the project to communicate with the database installed in the system.

After analyzing and studying the different programming languages, the application has been developed and installed on a web server. Then, a viability analysis of the system has been made, with some users' help who have tested the application.

Keywords: VoiceXML, Dialog systems, Voxeo, Lottery and bet, Speech interaction, Information.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Introducción y objetivos

En este primer capítulo se da a conocer el alcance del Proyecto Fin de Carrera propuesto, es decir, cada uno de los objetivos que se desea conseguir a lo largo de la investigación, estudio y desarrollo del mismo. Además de todo esto, se muestra una pequeña estructura del documento presentado, la cual, hace que sea más sencilla su comprensión.

1.1. Introducción

Este proyecto tiene como alcance facilitar el acceso a cierto tipo de información a personas que, desde una web les resultaría dificultoso o, en algunos casos, incluso imposible. La información obtenida, en particular, son los resultados de la lotería nacional.

La característica principal de este proyecto es la forma en la que vamos a ofrecer esta información. Vamos a establecer un contestador virtual, el cual va a gestionar cada una de las llamadas que reciba para entregar la información pedida por teléfono. De esta manera estamos aportando a la sociedad otra forma diferente de recibir información.

Una de las formas de comunicación más eficientes y naturales que poseemos es el habla. Esa es la motivación principal para realizar este proyecto, aprovechar una de las características más especiales que poseemos para interactuar con un sistema informático.

Hablar es muy dinámico e intuitivo, no hace falta enseñar a nadie para poder obtener esta información, basta con llamar y pedirlo. Con esto acercamos a diferentes grupos de personas la forma de conseguir información, por ejemplo, es complicado ver a una persona anciana utilizar el ordenador, pero, por el contrario, es muy habitual ver a una persona anciana usar el teléfono.

Además de este grupo social, podemos incluir otro grupo de personas que se verían gratamente beneficiados por los sistemas de diálogo, los invidentes. Según la organización mundial de la salud, se estima que en todo el mundo hay entre 40 y 45 millones de invidentes y 135 millones de personas con baja visión, para todos ellos,

cualquier sistema de diálogo que permita interactuar con ellos es una fuente de información muy preciada.

En los últimos tiempos estamos asistiendo a una explosión de los servicios ofrecidos por vía telefónica en base a sistemas de reconocimiento de voz IVR (Interactive Vocal Response) y sistemas de gestión de diálogo. Cabe pensar que es debido a que es mucho más fácil, rápido y todo el mundo lo puede utilizar en cualquier momento. En España, en total, hay 44,3 millones de líneas de teléfono declaradas, lo cual hace que cada español tenga un teléfono móvil en su propiedad y a su vez un dispositivo desde el cual se pueda obtener información.

Para la implementación de los sistemas de diálogo, el W3C **[W3C]** propone el estándar VoiceXML (Voice Extensible Markup Language) **[VOICEXML]**. Este lenguaje de programación permite interactuar entre una máquina y una persona, haciendo posible el entendimiento bidireccional, reconocimiento de voz e interpretación vocal de los símbolos.

Por lo tanto, el presente proyecto fin de carrera se basa en este estándar, dando vida a un sistema de diálogo capaz de dar la información actualizada de todos los sorteos o juegos realizados en el estado español **[LAE]**.

Una gran ventaja es poder acceder a estos servicios de forma no presencial. Gracias a los avances tecnológicos, se han creado y han ido evolucionando distintos canales que proporcionan estas prestaciones. La página web es uno de los medios más utilizados, pero como se ha comentado anteriormente, el acceso puede ser complicado para ciertos tipos de personas, mientras que con el sistema que se propone, esta accesibilidad se facilitaría.

1.2. Objetivos

El objetivo fundamental de este proyecto fin de carrera es el de realizar un sistema informático, el cual sea capaz de interpretar las peticiones por voz de un usuario y devolver la información exacta pedida previamente.

En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Realizar un estudio completo de los sistemas de diálogo, desde el concepto de interfaz oral a su arquitectura de módulos, pasando por los distintos criterios para su clasificación.

- Realizar un estudio detallado del estándar VoiceXML, así como de todos los lenguajes necesarios para implementar las distintas funcionalidades del sistema.

- Facilitar el uso de estos servicios a usuarios que desconocen tanto el uso de un ordenador o de Internet, por ejemplo, personas ancianas que están acostumbradas únicamente al uso del teléfono.

- Facilitar el acceso a las funcionalidades ofrecidas en esta aplicación a personas con discapacidades motoras o visuales, o a personas mayores con capacidades disminuidas, contribuyendo a hacer el mundo tecnológico más accesible para ellas.

1.3. Fases del desarrollo

A continuación se definen las fases del desarrollo que se han realizado a la hora de la finalización del proyecto. El proyecto se puede dividir en cuatro fases bien diferenciadas que se describen a continuación.

Fase 1: Investigación.

- **Estudio de los sistemas de diálogo:** Estudio tanto del funcionamiento, como de la estructura general de un sistema de diálogo bien definido.

- **Estudio de las tecnologías:** Se analizan tanto los lenguajes utilizados como las herramientas necesarias para hacer que el sistema funcione.

- **Toma de requisitos:** Se realiza el primer acercamiento al sistema que queremos generar. Se determinan todas las funcionalidades que queremos que realice el portal de voz.

- **Viabilidad del sistema:** Una vez definidos todos los requisitos, se realiza el análisis de la posible viabilidad del sistema, asociándolo a algún portal Web. Dado que no todos los portales Web son idóneos, seleccionamos dos: [ONCE], [LAE].

Fase 2: Preparación de entornos.

- **Estudio de las herramientas:** Se realiza un análisis de las herramientas posibles para poder crear un entorno de desarrollo local. En este caso definimos tres herramientas. Apache, MySQL y DUC.

- **Montaje del entorno de desarrollo:** Una vez seleccionadas las herramientas, procedemos al montaje de las mismas en una máquina local. Para realizar este montaje se utiliza una herramienta externa llamada XAMPP.

- **Pruebas unitarias del entorno:** Después del montaje del entorno, se procede a realizar pruebas unitarias para comprobar su correcto funcionamiento.

Fase 3: Desarrollo.

- **Análisis de los requisitos:** Se realiza un análisis exhaustivo de los requisitos y se divide el proyecto en diferentes módulos funcionales.

- **Diseño funcional:** Una vez realizada la definición de los módulos funcionales, se procede a definir cada uno de los submódulos pertenecientes al sistema diseñado.

- **Establecimiento de tareas:** El tercer paso en el desarrollo es la definición de las tareas dependientes de cada uno de los submodulos. Cada submódulo se divide en pequeñas tareas bien diferenciadas para realizar un desarrollo más sencillo y eficaz.

- **Desarrollo de la aplicación:** Se procede de desarrollar cada una de las tareas definidas en el punto anterior.

- **Pruebas unitarias:** Se realizan pruebas unitarias de cada funcionalidad por separado. Dado que se desea que el proyecto sea lo más estable posible, se establece una batería de pruebas unitarias.

- **Evaluación de la aplicación:** Se generan encuestas de evaluación que son entregadas a diversos usuarios, los cuales deben rellenarlas después de interactuar con el sistema. Se procede con el análisis de los resultados obtenidos.

Fase 4: Documentación.

- **Memoria del Proyecto Final de Carrera**: Redacción del presente documento de memoria del Proyecto Final de Carrera.
- **Creación de la presentación**.

A continuación se muestra un diagrama WBS (Work Breakdown Structure), donde se detallan en forma de árbol jerárquico todas las fases del desarrollo definidas, construido con la finalidad de ordenar de acuerdo a cierta lógica, las tareas temporales referentes e implicadas en la realización del desarrollo. Como se puede observar en la Figura 1.1, se divide en cuatro grandes módulos y cada uno de ellos en una sucesión de tareas que pertenecen a la misma familia.

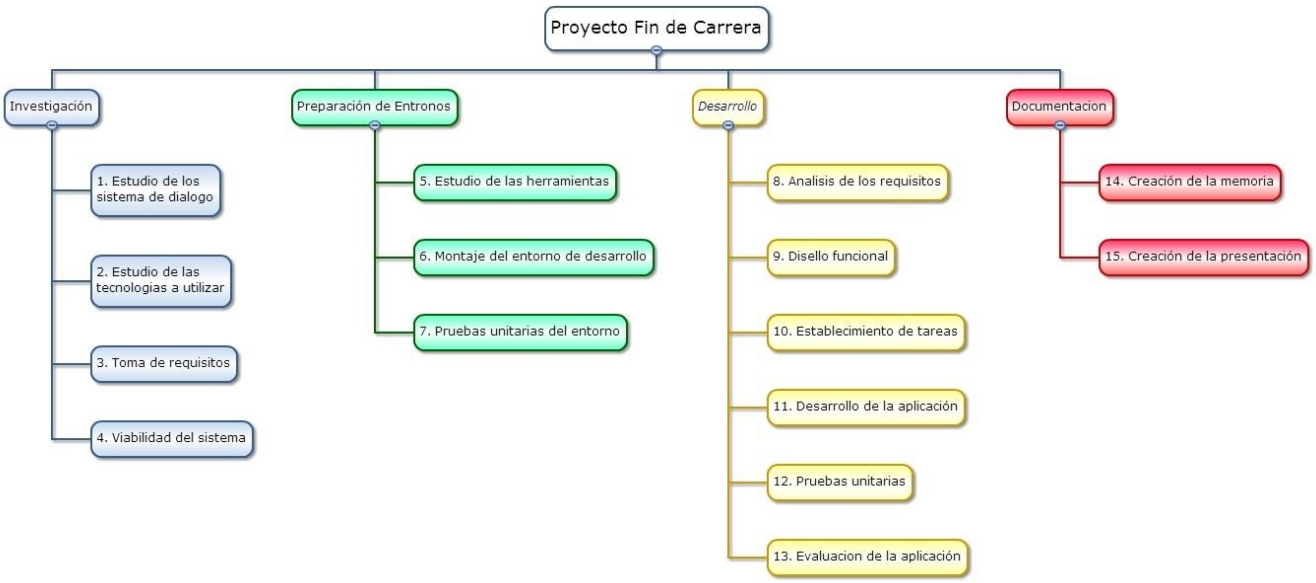

FIGURA 1.1.1 DIAGRAMA WBS REPRESENTANDO LOS MÓDULOS DEFINIDOS.

1.4. Medios empleados

A continuación se definen los medios empleados para el desarrollo de este Proyecto Fin de Carrera.

- Hardware**:**
    - Ordenador Sobremesa.
    - Micrófono.
    - Altavoces.
    - Periféricos utilizados: teclado y ratón.
- Software:
    - Editor de texto Notepad ++.
    - Editor de texto JVoxEdit
    - Microsoft Office 2010.
    - Aplicación de comunicación Skype.
    - Servidor independiente de plataforma XAMPP
    - WBS Editor.

En cuanto a la documentación empleada, se han utilizado gran cantidad de libros y artículos relacionados con los sistemas de diálogo, todos y cada uno de ellos queda reflejado de forma detallada en el apartado de Bibliografía.

1.5. Estructura de la memoria

Capítulo 1: Introducción y objetivos:

En este capítulo se realiza una explicación general de la situación actual de los sistemas de diálogo, se pretende estudiar la situación y ver de qué manera se puede plantear el desarrollo del proyecto. Además en este capítulo se establecen los objetivos a alcanzar y las fases de desarrollo a realizar.

Capítulo 2: Estado del arte:

En el estado del arte se hace un estudio exhaustivo del lenguaje utilizado en este proyecto fin de carrera, se define detalladamente el estándar VoiceXML y se realiza un estudio detallado de las herramientas presentadas por la plataforma utilizada.

Capítulo 3: Descripción general del sistema:

En este capítulo se explica de una manera general todo lo relacionado con el sistema, se muestran las bases seguidas para el desarrollo del mismo, se explican las herramientas utilizadas así como las tecnologías necesarias para la ejecución del mismo.

Capítulo 4: Descripción detallada del sistema.

Este capítulo explica de una manera detallada cada uno de los módulos de la aplicación. El capítulo se divide en submódulos, estos submódulos van recorriendo las diferentes funcionalidades del sistema para ir definiendo el sentido con el que fueron creados y como se ha conseguido dicha funcionalidad.

Capítulo 5: Evaluación del sistema.

El capítulo cinco es el encargado de explicar los pasos seguidos para realizar la evaluación del sistema. Está dividido en dos módulos, el primero define como se preparó la encuesta y que finalidad se quería buscar con las respuestas de los usuarios. El segundo obtiene las conclusiones obtenidas de las respuestas de los usuarios.

Capítulo 6: Conclusión y trabajo futuro

Este capítulo, muestra las conclusiones obtenidas del desarrollo del proyecto, se define si realmente el proyecto desarrollado cumple con los objetivos o requerimientos planteados para el sistema, además plantea un posible camino para realizar un proceso de mejora de la aplicación.

Presupuesto:

Se realizar un análisis de los costes. Estudiando recursos utilizados y tareas pertenecientes al desarrollo del proyecto. Con el esfuerzo realizado y los recursos utilizados se genera un presupuesto del proyecto fin de carrera planteado.

Glosario:

Este apartado muestra un catálogo de las palabras y expresiones de los textos que son difíciles de entender junto con el significado de cada uno de ellos.

Bibliografía:

Este apartado presenta un conjunto de referencias sobre publicaciones que se han ido consultando durante el desarrollo del proyecto presentado.

Capítulo 2

2. Estado del arte

En este capítulo procedemos a la explicación de lo que son los sistemas de diálogo, la estructura que se debe seguir a la hora de su creación, las herramientas utilizadas para la generación de este proyecto y el estudio completo del lenguaje estándar de programación VoiceXML que existe actualmente. Además de todo esto, explicaremos también las características y configuración de la plataforma usada en este proyecto, **Voxeo.**

2.1. Sistemas de diálogo

2.1.1. Introducción

Llamamos sistema de diálogo a cualquier procedimiento utilizado para facilitar la comunicación entre una persona y un sistema informático. El sistema de diálogo puede enmarcarse en el campo de la comunicación entre personas y ordenadores (HCI, Human-Computer Interaction). A día de hoy, muchas empresas utilizan este tipo de sistemas para ofrecer un servicio de información u otros servicios de forma automática, como puede ser concretar una cita médica o realizar una reserva de billetes de avión.

Actualmente la tecnología del habla se encuentra en un estado avanzado de madurez. Los sistemas que están basados en tecnología del habla están presentes en muchos ámbitos, a modo de ejemplo, podemos encontrar interfaces vocales en el sector de las telecomunicaciones, en los dispositivos portátiles, en la industria de automoción, en el acceso a la información facilitada por los medios de comunicación de masas. Un claro ejemplo de sistemas basados en el habla son los también conocidos como IVR (Interactive Voice Response systems), que son todos aquellos que permiten difundir información a través de la línea telefónica.

Además, con estos sistemas se facilita la comunicación en aquellos casos en los que no es posible realizar dicha comunicación con los métodos tradicionales. Dícese de la utilización de un teclado, un ratón o el monitor de un ordenador. Incluso podemos decir que podemos dar la posibilidad a personas con discapacidad motriz o visual de interactuar con un sistema en el que normalmente no son capaces de realizarlo. Tal y como se ha dicho anteriormente, una persona con discapacidad visual, puede interactuar con un sistema de diálogo para pedir su cita médica.

Definiríamos un sistema de diálogo ideal a aquel que sea capaz de reconocer el habla espontánea, comprender enunciados sin restricciones de contenido, proporcionar respuestas con sentido y gramaticalmente bien formadas, responder con voz completamente natural y que sea un sistema multimodal **[LLI06].**

Los sistemas de diálogo actuales están sujetos a limitaciones. Podríamos decir que son limitadas al reconocimiento automático del habla. Tanto la comprensión como la respuesta están restringidas a dominios específicos. Están condicionados por la naturalidad del habla sintetizada. Además de la necesidad de estrategias de verificación.

2.1.2. Metodología de los sistemas de diálogo

Construir una aplicación informática que pueda mantener una conversación con una persona de manera natural sigue siendo hoy en día un reto, dada la gran cantidad de fuentes de conocimiento que son necesarias y las limitaciones de las tecnologías utilizadas para obtener información del usuario **[GCL+09]**. Aun así, los avances en la investigación de las tecnologías del habla, nos permiten interactuar con cierto grado de flexibilidad.

A continuación, mostramos un diagrama de acciones de un sistema de dialogo, encargado de cumplir la finalidad para la que fue diseñado, Figura 2.1 [Gri07].

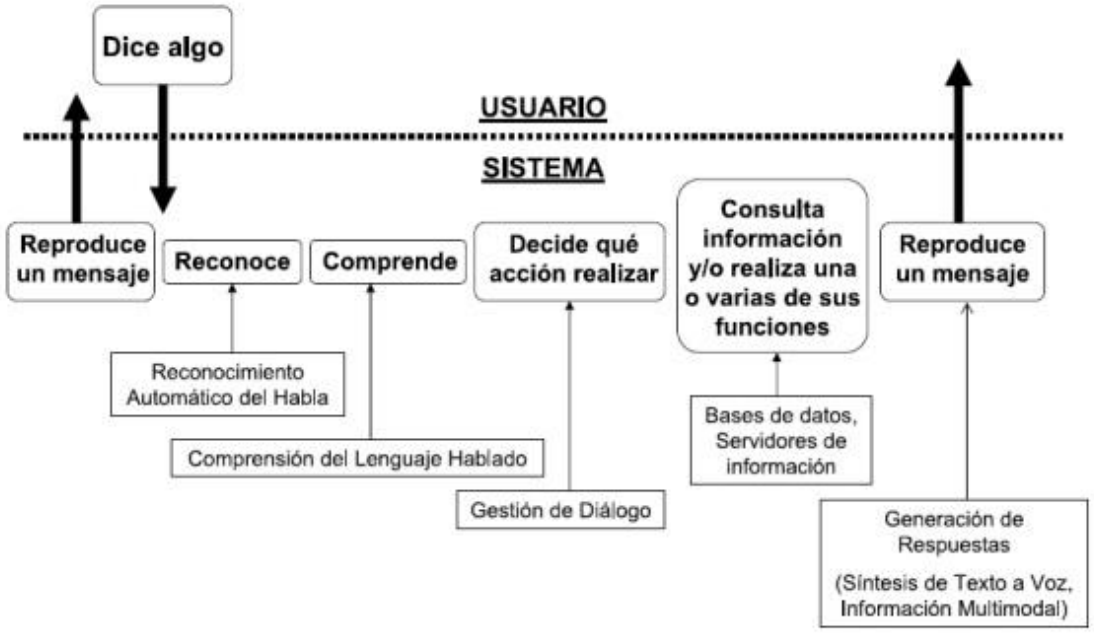

FIGURA 2.1 DIAGRAMA DE ACCIONES DE UN SISTEMA DE DIÁLOGO

Como se puede observar en el diagrama, el sistema reproduce un mensaje de bienvenida o bien un mensaje para mostrar al usuario las diferentes características del sistema. Una vez el usuario interactúa con el sistema, este debe realizar una secuencia de acciones:

1. Reconocer las palabras introducidas por el usuario.

2. Descifrar las palabras introducidas para intentar buscar un "significado útil" dentro de su sistema.

3. Realizar operaciones de acceso a base de datos u otros recursos del sistema, en los que se almacena la información que solicita el usuario o se registran las operaciones que desea conocer.

4. Se define la respuesta que debe tomar el sistema, es decir, indicar las acciones que debe de tomar el sistema después de cada solicitud del usuario.

5. Se reproduce el mensaje con la acción que ha seleccionado el usuario.

Debido al gran número de operaciones que el sistema debe ejecutar, es muy recomendable realizar un desarrollo modular, lo que permite dividir las dificultades entre los diversos componentes del sistema. A continuación mostramos un listado de los diferentes módulos:

- **Módulo de Reconocimiento Automático del Habla**. Realiza un estudio de la entrada al sistema y proporciona una secuencia más que probable de las palabras reconocidas.

- **Módulo de Comprensión del Habla**. A partir de la secuencia de palabras reconocidas, el sistema obtiene una representación semántica de su significado.

- **Gestor de Diálogo**. Considera la interpretación semántica de la petición del usuario. Se realiza un estudio más detallado en los siguientes puntos (**2.1.3 y 2.1.4**).

- **Módulo de Consulta a la Base de Datos de la Aplicación**. Recibe peticiones de consulta a la base de datos por parte del gestor de diálogo, las procesa y devuelve el resultado al gestor.

- **Módulo de Generación de Respuestas**. Recibe la respuesta del sistema y tiene como función generar un mensaje de respuesta formal, gramaticalmente correcto, que transmita el significado generado por el gestor de diálogo.

- **Sintetizador de Texto a Voz**. Componente del sistema que recibe la secuencia de palabras a transmitir y genera su correspondiente señal de audio.

A continuación se muestra un diagrama de la arquitectura modular para el desarrollo de un sistema de diálogo hablado [Gri07].

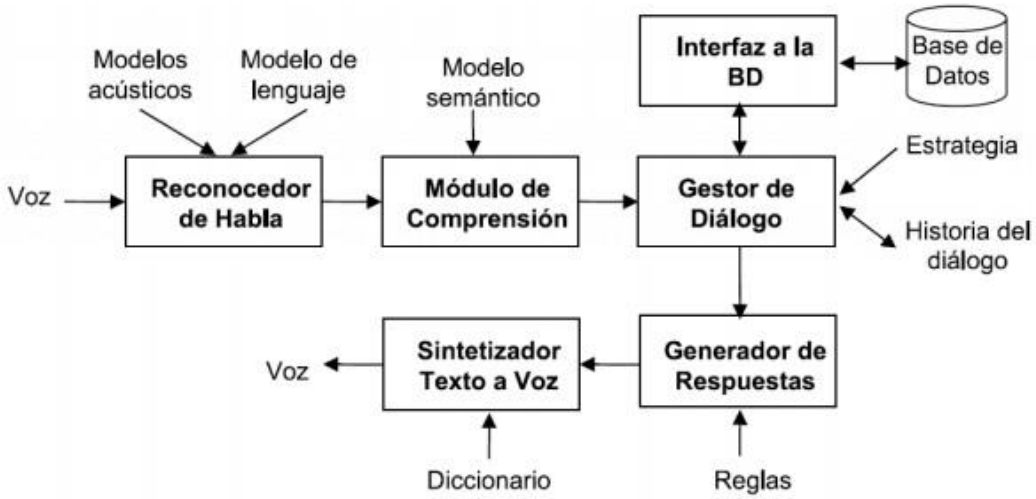

FIGURA 2.2 ARQUITECTURA MODULAR DE UN SISTEMA DE DIÁLOGO

2.1.3. La Gestión del diálogo

Uno de los principales problemas de los sistemas de diálogo es la identificación de los objetivos de diálogo del usuario **[ZSG+97].** A diferencia de lo que ocurre con las técnicas de reconocimiento del habla, en las que se busca una correcta transcripción de las frases pronunciadas por el usuario **[NEO99],** una buena gestión del diálogo permite solo quedarnos con aquellas palabras que realmente nos den la clave sobre la información recibida.

Tradicionalmente, la función del módulo de comprensión se limita a la extracción del marco semántico de la frase actual **[War94].** Este módulo es el encargado de recibir la salida del módulo de reconocimiento del habla y a su vez, devolver su salida al módulo de control del diálogo, dejando a este último el trabajo de interpretación y decisión de cómo continuar el diálogo.

La representación del significado de una frase puede hacerse mediante conceptos que determinan el tipo de información del turno de diálogo del usuario y mediante valores que proporcionan la información de la frase **[FCM+00]**. Un acercamiento puede ser representar el comportamiento del diálogo mediante actos de diálogo. Podríamos descomponer esos comportamientos en tres niveles:

1. **"Objetivos de diálogo".** Se pueden definir como el conjunto de acciones (i.e. verbos) que representan el carácter general del turno de diálogo y que responden a la intención del usuario. Por ejemplo, para la frase "enciende la luz del salón", un posible objetivo podría definirse como "encender (Luz)".

2. **"Atributos".** Representan el tipo de información que es necesaria dentro del dominio de aplicación y que es aportada por el usuario a través de la frase pronunciada. Para la frase del ejemplo anterior, esta quedaría recogida por "Atributo: Luz" en referencia al parámetro que deseamos modificar.

3. **"Valores".** Son los diferentes atributos. Una vez más, para el ejemplo anterior, resultaría "Valor: Salón".

2.1.4. Tipos de sistemas de gestión de diálogo

Los sistemas de gestión de diálogo se construyen siguiendo diversas filosofías **[GTH+05].** Se pueden clasificar dichas filosofías en varias alternativas:

- Se definen tres estrategias o soluciones para la gestión del diálogo distintas **[MCT02]:**

- Sistemas de estados finitos. Se genera un autómata de estados finitos, de este modo la estructura del diálogo se representa en forma de red de transiciones entre los distintos estados que lo describen.

- Sistemas basados en marcos. A diferencia de los anteriores, los sistemas basados en marcos definen el diálogo como una tarea análoga a la de completar o rellenar formularios, por medio de la cual una cantidad de información debe ser recuperada.

- Sistemas basados en agentes. Estos sistemas están basados en un proceso de colaboración entre agentes inteligentes, la principal ventaja de este sistema es que es capaz de hacer frente a diálogos más complejos. Los sistemas basados en este enfoque hacen uso de un modelo de colaboración para establecer la información mutua (es decir, conocimiento compartido). Así, los agentes colaboran para construir un modelo de la conversación y la información compartida a través de un conjunto de actos de diálogo dependientes e independientes de dominio [Novick and Ward, 1993].

- Por otra parte **[XXH+02]** lleva a cabo una clasificación basada en cuatro categorías:

  - DITI (modelo del diálogo implícito, modelo de la tarea implícito) que correspondería, por ejemplo, a los modelos de estados finitos.

  - DITE (modelo del diálogo implícito, modelo de la tarea explícito) que correspondería a los modelos basados en marcos.

  - DETI (modelo del diálogo explícito, modelo de la tarea implícito).

  - DETE (modelo del diálogo explícito, modelo de la tarea explícito).

Resulta muy complicado realizar una clasificación que se adecue a todos los tipos de gestión de diálogo, ya que todos estos enfoques no son excluyentes y normalmente se utilizan de forma conjunta.

2.1.5. Clasificación de los sistemas de diálogo

Los sistemas de diálogo se pueden clasificar mediante la forma de interacción entre el usuario y el sistema en cuestión.

- **Sistemas de diálogo guiados por la máquina**. Estos sistemas están basados en las restricciones de la máquina frente al usuario. Se establece una

comunicación pregunta-respuesta en la cual el usuario debe preguntar lo que el sistema es capaz de responder.

- **Sistemas de diálogo cooperativos**. Aceptan las interrupciones y negociaciones por parte del usuario, reparten de forma equilibrada el turno de palabra e incorporan mecanismos de detección de incoherencias gramaticales.

- **Sistema de diálogos guiados por el usuario**. El sistema es capaz de aprender nuevas estrategias comunicativas en función del comportamiento del usuario. El usuario lleva la iniciativa de diálogo en cualquier instante de la interacción, se le permite que utilice lenguaje natural y no se delimitan los mensajes del sistema.

Además de la clasificación anterior, los sistemas de diálogo se pueden clasificar de la siguiente manera **[Gri07]:**

Según la dependencia del hablante:

- **Dependiente del hablante.** Sistema definido para trabajar con un único hablante. Estos sistemas son los menos flexibles, pero como contrapunto, son fáciles de desarrollar, menos costosos y más flexibles**.**

- **Independientes del hablante.** Sistema definido para trabajar con un mismo grupo de habla parlante. Por ejemplo, en España, la mayor parte de la población habla castellano, pero no todos tienen el mismo acento. Estos sistemas son más complicados de implementar pero son mucho más flexibles.

Según el tipo de comunicación:

- **Unimodales.** Son aquellos sistemas que únicamente utilizan el habla.

- **Multimodales.** Son aquellos sistemas que utilizan varios canales de comunicación para proporcionar las entradas y las salidas.

Según el tipo de idioma:

- **Monolingüe.** Sistema que únicamente permite la interacción con un único idioma.

- **Multilingüe.** Sistemas que soportan la interacción con varios idiomas.

Según el tipo de discurso:

- **Reconocimiento de palabras aisladas.** El usuario se comunica con el sistema utilizando palabras individuales (o frases) tomadas de un vocabulario

determinado. El reconocimiento se lleva a cabo comparando el patrón acústico de la palabra a reconocer con los patrones almacenados.

- **Reconocimiento de palabras conectadas.** El usuario se comunica de una forma fluida con una sucesión de palabras pertenecientes a un vocabulario restringido. El reconocimiento se lleva a cabo basándose en la coincidencia de palabras de referencia aisladas.

- **Reconocimiento contínuo.** El usuario se comunica hablando fluidamente usando palabras de vocabulario que están conectadas y no están separadas por pausas. El lenguaje continuo es más difícil de tratar debido a la variedad de efectos.

- **Reconocimiento discreto.** El usuario se comunica con palabras simples, necesitando de una pausa entre la pronunciación de cada una de ellas. Ésta es la forma más sencilla de reconocimiento, debido a que resulta muy sencillo reconocer los puntos de finalización.

Según la adaptación:

- **Adaptación. Sistemas adaptativos:** El sistema es capaz de aprender nuevas estrategias comunicativas en función del comportamiento del usuario.

2.1.6. Sistemas de Diálogo Existentes: Ejemplos representativos

En las siguientes líneas se presentan algunas de las aplicaciones de distintos ámbitos cuya interfaz está diseñada para interactuar con el usuario a modo de sistema de diálogo oral.

Información.

A continuación mostramos ejemplos de aplicaciones encargadas de dar información al usuario.

ARISE (Automatic Railway Information Systems for Europe). Sistema desarrollado para ofrecer información sobre viajes en tren. El sistema es capaz de dar información sobre los horarios de salida y llegada de los trenes según las especificaciones de los usuarios vía telefónica **[ARISE].**

RAILTEL (Railway Telephone Information Service). Sistema desarrollado por CSELT (Italia), un sistema parecido al descrito anteriormente. Los usuarios se comunican

con este sistema para obtener información sobre viajes/horarios de los trenes **[RAILTEL].**

MASK (Multimodal-Multimedia Automated Service Kiosk). Sistema desarrollado por el Spoken Language Processing Group (Francia). Proporciona horarios de tren, reservas, precios, etc. Los usuarios se pueden comunicar con el sistema mediante habla y una pantalla sensible al tacto, mientras que éste se puede comunicar con los usuarios mediante habla, gráficos, vídeo y sonido **[MASK].**

ATIS (Air Travel Information System), sistema encargado de dar información a los usuarios sobre los horarios y tarifas de los vuelos **[ATIS]**.

PEGASUS. Información de viajes en avión. EE.UU. **[PEGASUS]**.

VOYAGER. Sistema encargado de dar información sobre regiones geográficas de Cambridge, Massachusetts en EEUU. El usuario puede pedir información sobre las distancias, tiempo en viajes o direcciones sobre establecimientos alojados en esta área **[VOYAGER]**.

JUPITER. Se trata de un sistema de diálogo oral diseñado para proporcionar al usuario la información meteorológica actualizada día a día mediante vía telefónica. Ha sido desarrollado por el Instituto Tecnológico de Massachusetts (MIT), en Estados Unidos **[JUPITER].**

TELLCORREO. Se trata de un sistema de diálogo multilenguaje que permite a los usuarios la lectura de correos electrónicos **[TELLCORREO]**.

Reserva.

A continuación, mostramos algunos sistemas de diálogo que aparte de dar información al usuario, también son capaces de realizar reservas.

MERCURY. Se trata de una interfaz oral basada en el uso telefónico que proporciona información acerca de vuelos y sus precios. Mercury permite a los usuarios reservar vuelos a más de 200 destinos en Estados Unidos y en el resto del mundo **[MERCURY].**

VICO (Virtual Intelligent Co-Driver). Permite la interacción natural entre los usuarios y dispositivos digitales. El sistema de diálogo está basado en el ámbito del automóvil y proporciona a los usuarios una interfaz usable con lenguaje natural así como servicios innovadores, como pueden ser navegación, planificación de rutas, posibilidad de reservar hoteles o diseñar rutas turísticas.

Educación.

También existen aplicaciones que dan servicio de ayuda al usuario. Se están desarrollando aplicaciones capaces de ayudar en la educación del usuario, es decir, ayudar a incrementar el desarrollo personal de cada individuo que lo usa.

ITSPOKE. Se trata de un sistema de diálogo desarrollado para el ámbito académico. El sistema conversa con los estudiantes y les proporciona información, así como correcciones de sus errores de aprendizaje **[ITSPOKE]**.

VOCALIZA. Sistema de diálogo que se utiliza para terapias de voz en español y ayuda a los terapeutas a tratar a pacientes con distintas patologías y carencias en su habilidad lingüística.

Transacciones.

Además de lo anteriormente enumerado, los sistemas conversacionales también pueden llevar a cabo transacciones. Son las aplicaciones relacionadas con la banca o comercio electrónico. A continuación mostramos algunos ejemplos.

YDILO. Sistema de conversación encargado de gestionar todo lo relacionado con la venta de entradas al usuario.

CINES VERBIO. Simula un sistema automático de reserva de entradas de cine. Está implementado con el lenguaje de programación VoiceXML (al igual que el proyecto en el que está basado esta memoria) **[VERBIO]**.

BBVA. Sistema que permite solicitar transferencias o domiciliaciones, vender o comprar valores, contratar seguros o realizar aportaciones a planes de pensiones o fondos de inversión. Se comprueba la identidad del cliente mediante una clave personal de acceso **[BBVA].**

2.2. Voice Extensible Markup Language (VoiceXML)

2.2.1. Introducción al estándar VoiceXML

El origen de VoiceXML data de 1995. VoiceXML es el estándar propuesto por W3C (Consorcio World Wide Web) para la implementación de sistemas de diálogo. Fue diseñado como un lenguaje de diálogo basado en XML (eXtensible Markup Language), que

pretendía simplificar el proceso de desarrollo de aplicaciones de reconocimiento de voz dentro de un proyecto de AT&T llamado PML (Phone Markup Language).

En 1998, W3C, organizó una conferencia sobre navegadores por voz. Motorola e IBM habían desarrollado sus propios lenguajes mientras que AT&T y Lucent tenían diferentes versiones de su PML original. Muchos otros asistentes también estaban diseñando lenguajes parecidos. Debido a toda esta situación AT&T, IBM y Motorola formaron el foro de VoiceXML para unificar esfuerzos.

En Marzo del año 2000, se publicó la versión 1.0 de VoiceXML. Fue publicada por el foro VoiceXML. Posteriormente, el Fórum entregó el control del estándar al W3C, y ahora es éste quien se encarga de garantizar el cumplimiento de la normativa legal, así como de los temas educativos y de marketing. Desde entonces, el W3C ha publicado la versión 2.0 de VoiceXML y la Candidate Recomendation para la versión 2.1. El working draft de VoiceXML 3.0 fue aprobado en diciembre de 2010.

VoiceXML es un lenguaje diseñado para crear sistemas de diálogo mediante la voz que incluyen voz sintetizada, reconocimiento de entrada por voz o DMTF, grabación de diálogos, funciones de telefonía e iniciativa mixta. Su objetivo principal es usar las ventajas y características del desarrollo web en la implementación de aplicaciones de voz interactivas.

Seguidamente, se muestra un pequeño ejemplo de código VoiceXML.

```
<?xml version="1.0" xml:lang="es-ES" encoding="UTF-8"?>
<vxml version="2.0">
  <form>
    <block>
      <prompt>
        ¡Hola Mundo!
      </prompt>
    </block>
  </form>
</vxml>
```

FIGURA 2.3 EJEMPLO 'HOLA MUNDO' EN VOICEXML

Una arquitectura definida para una aplicación desarrollada en VoiceXML tiene cuatro componentes: servidor de documentos, intérprete de contexto VoiceXML, intérprete VoiceXML y una plataforma de implementación.

- **Plataforma de Implementación:** Está controlada por el intérprete de contexto VoiceXML. Es la encargada de generar los eventos en respuesta a las acciones del usuario y a los eventos del sistema. Como se puede ver en el esquema, estos eventos están gestionados por dos módulos, o el intérprete de VoiceXML o el intérprete de contexto de VoiceXML.

- **Servidor de documentos:** Es el módulo encargado de procesar las peticiones enviadas por el intérprete de VoiceXML. Además también genera los documentos asociados a la respuesta de la petición, las cuales son procesadas por el intérprete de VoiceXML.

- **Interprete de contexto:** El intérprete de contexto VoiceXML, monitorea las solicitudes del usuario, de manera paralela con el intérprete VoiceXML.

A continuación, se puede observar un esquema del modelo a seguir.

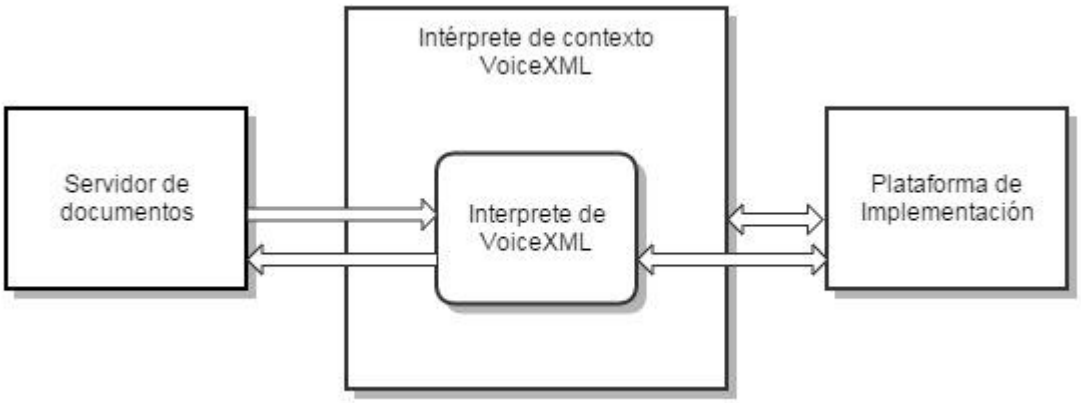

FIGURA 2.4 MODELO DE ARQUITECTURA VOICEXML

El objetivo principal de VoiceXML es llevar todo el potencial del desarrollo web existente a las aplicaciones basadas en sistemas de diálogos. Por lo tanto este lenguaje permite la integración de estos servicios de voz con servicios de datos. Podemos definir un servicio de voz como una secuencia de diálogo interactiva entre un usuario y una plataforma de implementación.

Por tanto, VoiceXML es el estándar que permite el desarrollo de aplicaciones de voz, y ofrece las siguientes ventajas:

- Minimiza las interacciones cliente/servidor mediante múltiples interacciones por documento XML.

- Separa la interacción del código de usuario (en VoiceXML) de la lógica de servidor.

- Es una tecnología independiente de la plataforma.
- Permite la portabilidad y transferencia de datos entre aplicaciones heterogéneas.
- Facilita la labor de los desarrolladores, ya que no necesitan conocer los detalles de implementación.
- Reutiliza eficientemente la infraestructura existente.
- Rapidez de desarrollo.
- Portabilidad sencilla entre diferentes plataformas hardware y software.
- Flexibilidad: al ser un estándar abierto, permite la interoperabilidad de tecnologías y una amplia selección de aplicaciones VoiceXML.

2.2.2. Conceptos básicos

Un documento VXML se puede definir como una máquina de estados finitos. El usuario se encuentra siempre en un estado o diálogo en cada momento. Cada diálogo determina la transición al siguiente dialogo. Las transiciones se definen mediante URIs (Uniform Resource Identifier), las cuales definen el siguiente documento, y por lo tanto, el siguiente diálogo a utilizar.

La ejecución finaliza cuando no se especifica un sucesor o si se ejecuta algún elemento encargado de la finalización del diálogo.

2.2.2.1. Diálogos y Subdiálogos

Existen dos tipos de diálogos: formularios y menús.

- Los formularios son los encargados de interactuar y recolectar los valores de las variables del sistema. Cada campo puede especificar una gramática que defina las entradas correctas para el mismo.
- Los menús representan las diferentes opciones definidas y la correspondiente transición a los diferentes diálogos, todo ello dependiendo de las diferentes elecciones que realice el usuario.

Los subdiálogos se podrían definir como si fuese una llamada a una función. Éstos ofrecen un mecanismo para invocar una nueva iteración y volver al punto original del

formulario. Las variables de la instancia, gramáticas y la información del estado son guardados y están disponibles al volver al documento original. Los subdiálogos pueden ser utilizados, por ejemplo, para crear una secuencia de confirmación que requiere de un acceso a una base de datos.

2.2.2.2. Sesiones

Una sesión comienza cuando un usuario empieza a interactuar con el intérprete de contexto de VoiceXML, continúa mientras los documentos son cargados y procesados y finaliza cuando lo solicita el usuario, un documento o el intérprete de contexto.

2.2.2.3. Aplicación

Una aplicación es un conjunto de documentos que comparten el mismo documento raíz. Siempre que un usuario interactúe con un documento en una aplicación, el documento raíz de esta aplicación se carga también.

El documento raíz de la aplicación permanece cargado mientras que el usuario realice transiciones entre documentos que compartan dicho documento raíz y se descarga cuando se realiza una transición a un documento que no pertenece a la aplicación.

Mientras está cargado el documento raíz, las variables son compartidas y están disponibles para los otros documentos como variables de aplicación, y además, todas sus gramáticas permanecen activas.

2.2.2.4. Gramáticas

Cada diálogo puede tener asociadas una o más gramáticas de voz o DTMF.

En las aplicaciones dirigidas por máquinas, cada gramática está activa solamente cuando el usuario está en ese diálogo.

En las aplicaciones de iniciativa mixta, donde el usuario y la máquina se alternan en determinar qué hacer después, algunos de los diálogos están marcados para hacer sus gramáticas activas incluso cuando el usuario está en otro diálogo del mismo documento, o en otro documento en la misma aplicación. En estas situaciones, si el usuario dice algo que encaja con otra gramática activa de otro diálogo, se ejecuta una transición al mismo. Este tipo de iniciativas agregan flexibilidad y poder a las aplicaciones de voz.

2.2.2.5. Eventos

VoiceXML proporciona un mecanismo de rellenado de formularios para manejar las entradas “normales” del usuario. Además, VoiceXML define un mecanismo para manejar eventos no reconocidos por el formulario.

Los eventos son lanzados por la plataforma bajo una variedad de circunstancias, como que el usuario no responda, no responda de una forma inteligible o haga una petición de ayuda. El intérprete solo lanza los eventos si encuentra un error en la semántica de un documento VoiceXML. Los elementos pueden ser capturados para realizar alguna acción al respecto. El control de eventos comunes puede especificarse en cualquier nivel, y se aplica a todos los niveles inferiores.

2.2.2.6. Enlaces

Los enlaces soportan iniciativas mixtas. Éstos especifican transiciones a otros puntos del documento, otro documento dentro de la aplicación, o un documento de otra aplicación.

2.2.3. Elementos VoiceXML

En la siguiente tabla se describen los elementos pertenecientes a VoiceXML:

Elemento	Funcionalidad
<assign>	Asigna un valor a una variable
<audio>	Reproduce un clip de audio dentro de un mensaje.
<block>	Contenedor (no interactivo) de código ejecutable.
<catch>	Captura un evento.
<choice>	Define un elemento (ítem) del menú.
<clear>	Borra uno o más ítems del formulario.
<disconnect>	Desconecta una sesión.

Elemento	Funcionalidad
<else>	Elemento utilizado en la estructura <if>.
<elseif>	Elemento utilizado en la estructura <if>.
<enumerate>	Notación abreviada para enumerar las opciones en un menú.
<error>	Captura un evento error.
<exit>	Finaliza la sesión.
<field>	Declara un campo de entrada en un formulario.
<filled>	Acción a ejecutar cuando se rellenan los campos.
<form>	Cuadro de diálogo para presentar información y recopilar datos.
<goto>	Transición a otro diálogo en el mismo o diferente documento.
<grammar>	Especifica un reconocimiento de voz o una gramática DTMF.
<help>	Captura un evento ayuda.
<if>	Lógica condicional.
<initial>	Declara código inicial antes de entrar en un formulario (iniciativa mixta).
<link>	Especifica una transición válida para todos los diálogos en el ámbito de la aplicación.
<log>	Genera un mensaje de depuración.

Elemento	Funcionalidad
<menu>	Cuadro de diálogo para elegir entre varias alternativas.
<meta>	Define un elemento de metadata en formato nombre/valor.
<metadata>	Define información metadata mediante el esquema metadata.
<noinput>	Captura el evento noinput.
<nomatch>	Captura el evento nomatch.
<object>	Define extensiones a medida.
<option>	Especifica una opción en un <field>.
<param>	Parámetro en <object> o <subdialog>.
<prompt>	Salida de audio para el usuario.
<property>	Configuración de la plataforma de control de la aplicación.
<record>	Graba una muestra de audio.
<reprompt>	Reproduce un prompt cuando un campo es revisitado después de un evento.
<return>	Retorno desde un sub-diálogo.
<script>	Especifica un bloque de código ECMAScript.
<subdialog>	Invoca a un diálogo como un subdiálogo del actual.
<submit>	Presenta valores a otro documento del servidor.

Elemento	Funcionalidad
<throw>	Lanza un evento.
<transfer>	Transfiere la llamada a otro destino.
<value>	Inserta el valor de una expresión en un prompt.
<var>	Declara una variable.
<vxml>	Elemento de mayor jerarquía en cada documento VoiceXML.

TABLA 2.1: ELEMENTOS VOICEXML

A continuación, se muestra una tabla con los atributos de <VXML>.

Atributo	Descripción
version	Versión VoiceXML del documento (obligatorio).
xmlns	Espacio de nombres designado para VoiceXML (obligatorio).
xml:base	URI base para el documento. Todas las referencias relativas dentro del documento lo toman como base.
xml: lang	Identificador de idioma del documento. Si se omite, el valor es un valor predeterminado específico de la plataforma. Se hereda hacia niveles inferiores en la jerarquía del documento.
application	URI del documento raíz de la aplicación.

TABLA 2.2. ATRIBUTOS VOICEXML

2.2.4. Constructores de diálogo

2.2.4.1. Formularios

Los formularios son los elementos más importantes de un documento VoiceXML. Podemos decir que un documento se descompone en los siguientes ítems:

- Ítems del formulario. Elementos que son visitados en el bucle principal del algoritmo interpretado por el formulario.
- Declaración de variables.
- Manejadores de eventos.
- Proveer acciones. Bloques de procedimientos lógicos que son ejecutados cuando ciertas combinaciones de variables de entrada son asignados.

Los atributos de los formularios son:

Atributo	Descripción
Id	El nombre del formulario. Si está especificado, el formulario puede ser referenciado desde el propio documento o desde otro documento.
scope	Es el alcance por defecto de las gramáticas de los formularios. Si es un dialogo, la gramática del formulario esta activa durante cualquiera de los diálogos del mismo documento. Si el alcance es el documento y el documento es el documento raíz de la aplicación, entonces la gramática del formulario esta activa durante cualquier diálogo perteneciente a cualquier documento.

TABLA 2.3. ATRIBUTOS DEL FORMULARIO

2.2.4.1.1. Interpretación de los formularios

Los formularios son analizados por un implícito algoritmo de interpretación (FIA). La FIA tiene un bucle principal, con el cual va seleccionando los formularios y después los va visitando. El primer formulario seleccionado es el primero encontrado cuya condición de guarda no sea satisfecha.

Una interpretación de un formulario generalmente envuelve los siguientes acontecimientos:

- Seleccionar uno o más sonidos que reproducir.
- Recolectar la entrada del usuario si es correcta la guarda en uno de los ítems de entrada, si no lo es, lanzará algún evento.
- Interpretar alguna de las entradas recibidas en los ítems de entrada.

  La FIA termina cuando el intérprete transfiere el control a otro documento o cuando se recibe una salida implícita al no quedar ítems ilegibles en el documento.

2.2.4.1.2. Ítems de los formularios

Los ítems de los formularios son elementos que pueden ser visitados en el bucle del algoritmo de interpretación principal. Existen dos tipos de ítems: los ítems de entrada o los ítems de control.

A continuación, se procede a su explicación.

- **Ítems de entrada.** Un ítem de entrada especifica una variable recolectada desde el usuario. Los ítems de entrada tienen *<prompts>* que le dicen al usuario qué es lo que quieren que les diga o alguna clave para que el usuario sepa qué responder. Además, define la gramática que debe reconocer lo que el usuario diga, y sobre todo, algunos manejadores de eventos que son los encargados de procesar cualquier resultado de evento. También, un ítem de entrada debe tener un elemento *<filled>* que define la acción de recuperar la entrada del usuario y guardarlo en una variable.

Ítem	Descripción
<field>	Input de entrada cuyo valor se obtiene a través de gramáticas de voz y DTMF.
<record>	Input de entrada cuyo valor de entrada es un fragmento de audio que graba la entrada del usuario. (por ejemplo, un mensaje en un contestador de voz)
<transfer>	Input de entrada que realiza una transferencia a otro número de teléfono.

Ítem	Descripción
<object>	Este input de entrada invoca a un objeto de una plataforma específica a partir de varios parámetros. El objeto de retorno es un objeto ECMAScript.
<subdialog>	Input de entrada que invoca otro diálogo en el documento activo o invoca a otro documento VoiceXML. El objeto de retorno es un objeto ECMAScript.

TABLA 2.4 ATRIBUTOS DEL FORMULARIO

- **Ítems de control.** Podemos definir dos tipos diferentes de ítems de control:
  - **<Block>** Contiene un bloque de código a ejecutar.
  - **<Initial>** Controla la interacción inicial en un formulario de iniciativa mixta.

2.2.4.1.3. Variables y condiciones de los ítems de un formulario

Cada ítem del formulario tiene asociada una variable, la cual por defecto está sin definir cuando se entra en el formulario. Esta variable contendrá el resultado de la interpretación del ítem del formulario.

Cada ítem del formulario tiene asignado una condición de guarda, la cual rige cuando sí y cuando no los ítems del formulario pueden ser seleccionados por el algoritmo de interpretación del formulario.

A continuación, se muestra una tabla con los atributos que componen un ítem del formulario.

Atributo	Descripción
name	Nombre de la variable del ítem del formulario con ámbito de diálogo que contendrá el valor del ítem del formulario.
expr	El valor inicial de la variable del ítem del formulario. Su valor por defecto es undefined. Si se ha inicializado con un valor, entonces el

Atributo	Descripción
	ítem del formulario no se ejecutará a menos que el valor de esta variable se borre.
cond	Expresión a evaluar junto con la variable del ítem del formulario. Si no se especifica, por defecto será true.

TABLA 2.5. ATRIBUTOS DE LOS ÍTEMS DEL FORMULARIO

2.2.4.1.4. Formularios dirigidos

Los formularios más simples y los más comunes son los ejecutados exactamente una vez y en orden secuencial de implementación. Las gramáticas solo están activas en el estado visitado.

A continuación se muestra un ejemplo de este tipo de formularios. El sistema pregunta al usuario por su nombre, una vez reconocido, se lo repite.

```
<?xml version="1.0"?>
<vxml version="2.0">
   <form id="Nombre">
      <field name="nom" type ="nombre">
         <prompt>Digame su nombre.</prompt>
      </field>
      <filled>
         <block>
         <prompt>
            Usted se llama <value expr = "nom"/>
         </prompt>
         </block>
      </filled>
   </form>
</vxml>
```

FIGURA 2.5. EJEMPLO FORMULARIO DIRIGIDO

2.2.4.1.5. Formularios de iniciativa mixta

Para realizar un formulario de iniciativa mixta, tanto el sistema informático como el hombre, deben tener una conversación directa. Deben tener uno o más niveles de gramática. El dialogo se debe escribir en diferentes caminos. Si un formulario tiene diferentes niveles de gramática, los ítems de entrada pueden estar guardados en algún

orden y más de una variable de entrada puede ser completada como resultado de una simple entrada en el sistema.

Una estructura común en este tipo de formularios combina un elemento <initial> que solicita una respuesta general *<field>*, que a su vez solicita una información específica.

```
<form id="informacionVuelos">
<grammar src="ciudades.grxml"/>
   <block>
      <prompt bargein="false">
      Bienvenido al servicio de información Volar
      </prompt>
   </block>
   <initial name="inicio">
      <prompt>
      ¿Para qué ciudad y provincia desea conocer los vuelos?
      </prompt>
   <noinput>
      <assign name="inicio" expr="true"/>
   </noinput>
   </initial>
   <field name="provincia">
      <prompt>
      ¿qué provincia?
      </prompt>
   </field>
   <field name="ciudad">
      <prompt>
      Por favor diga la ciudad en
      <value expr="provincia"/>
      para la que desea conocer los vuelos disponibles.
      </prompt>
   <filled>
   <submit next="/vuelos" namelist="ciudad provincia"/>
   </filled>
   </field>
</form>
```

FIGURA 2.6 EJEMPLO FORMULARIO DE INICIATIVA MIXTA

2.2.4.2. Menús.

Un menú es una estructura de un formulario que contiene un único campo anónimo que obliga al usuario a realizar una petición de una elección o transición a diferentes sitios basados en la elección. Como en un formulario regular, cuando el usuario está ejecutando otro diálogo, éste permanece activo.

2.2.4.2.1. Elementos de menú.

El menú se compone de diferentes elementos, estos elementos determinan el alcance de la gramática. Los atributos de estos elementos se definen a continuación:

Atributo	Descripción
Id	Es el identificador del menú. Esto permite al menú ser el objetivo de las siguientes ordenes: <goto> y <submit>
Scope	Es el ámbito de la gramática perteneciente al menú.
Dtmf	Cuando este atributo está a verdadero, a las primeras nueve entradas que no tienen explícitamente definido un valor, son asignadas desde 1 hasta 9. Si la elección no se realizó no fue especificada se lanzará un error de tipo *error.badfetch*.
accept	Cuando está definido a *"exact"* (valor por defecto), el texto en el menú definido debe ser exactamente igual a la entrada recibida reconocida. Cuando se establece como *"approximate"* el texto de la elección definido debe aproximada a la frase reconocida.

TABLA 2.6. ELEMENTOS DEL MENÚ

2.2.4.2.2. Elementos <*choice*>

Los elementos <*choice*> tienen diferentes propuestas:

- Pueden especificar gramáticas de voz, sus propias gramáticas o alguna generada automáticamente por el sistema.
- Puede generar gramáticas de tipo DTMF.
- El contenido puede ser usado por el formulario para definir una serie de cadenas <*enumerate*>.
- Además, también son capaces de especificar eventos que pueden ser lanzados o las URL de transición cuando el usuario realice una elección.

Los atributos de estos elementos se definen a continuación.

Atributo	Descripción
dtmf	La secuencia DTMF para esta elección. Es equivalente a una gramática DTMF. A diferencia de las gramáticas DTMF, el espacio en blanco es opcional.
accept	Anula la configuración del accept en <menu> para esta selección particular. Para saber más sobre esta campo consultar la Tabla 2.6.
next	La URL del siguiente diálogo o documento.
expr	Especifica una expresión a evaluar como URI para la transición.
event	Especifica un evento para ser lanzado en vez de especificar una transición.
eventexpr	Expresión ECMAScript la cual evalúa el nombre del evento que será lanzado.
message	Cadena de mensaje que proporcionara el contexto sobre el evento que será lanzado.
messageexpr	Expresión ECMAScript la cual evalúa la cadena del mensaje.

TABLA 2.7. ATRIBUTOS DE LOS ELEMENTOS <CHOICE>

Exactamente uno de los siguientes elementos debe ser especificado: “next”, “expr”, “event” o “eventexpr”. Si por el contrario no se especifica ninguno, se lanzará un error: *error.badfetch*. Lo mismo ocurre con los campos “message” o “messageexpr” si no es declarado al menos uno de ellos.

Si un elemento de gramática es especificado en <choice>, la gramática externa es usada en vez de la gramática generada automáticamente. Esto permite al usuario obtener el control de las gramáticas seleccionadas.

2.2.4.3. DTMF en menús

Los menús pueden depender únicamente del habla, únicamente en DTMF, o combinando los dos tipos incluyendo un <property> en el elemento <menu>. Alternativamente, se puede establecer en el menú el atributo DTMF a verdadero para asignar secuencialmente dígitos DTMF para cada una de las primeras nueve elecciones posibles que no han sido especificadas en las secuencias DTMF.

Como puede observarse en el siguiente ejemplo, la primera elección tiene DTMF "1", y consecutivas.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu dtmf="true">
  <property name="inputmodes" value="dtmf"/>
    <prompt>
      For sports press 1, For weather press 2, For Stargazer astrophysics press 3.
    </prompt>
  <choice next="http://www.sports.example.com/vxml/start.vxml"/>
  <choice next="http://www.weather.example.com/intro.vxml"/>
  <choice dtmf="0" next="#operator"/>
  <choice next="http://www.stargazer.example.com/voice/astronews.vxml"/>
</menu>
</vxml>
```

FIGURA 2.7 EJEMPLO DE GRAMÁTICA EN FORMULARIO

2.2.4.4. Elementos enumerados

El elemento <enumerate> es una descripción generada automáticamente de las elecciones disponibles para el usuario. Este elemento, especifica una plantilla que es aplicada a cada una de las elecciones, en el orden en el que aparezcan en el menú. Si es usado sin contenido, se utilizará una plantilla por defecto. Esta plantilla por defecto es generada por el intérprete de contexto. Si tiene contenido definido, el contenido es el utilizado en la plantilla. Este especificador debe referirse a dos variables especiales: _prompt y _dtmf. La primera, establece el prompt elegido y la segunda, establece la elección del usuario a la secuencia DTMF.

Seguidamente, se muestra un ejemplo de estos elementos enumerados.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu dtmf="true">
    <prompt>
        Welcome home.
        <enumerate> For <value expr="_prompt"/>, press <value expr="_dtmf"/>.
        </enumerate>
     prompt>
   <choice next="http://www.sports.example.com/vxml/start.vxml"/>
   <choice next="http://www.weather.example.com/intro.vxml"/>
   <choice next="http://www.stargazer.example.com/voice/astronews.vxml"/>
</menu>
</vxml>
```

FIGURA 2.8 EJEMPLO DE LOS ELEMENTOS ENUMERADOS

En este ejemplo el menú promp debería ser:

C: Welcome home. For sports, press 1. For weather, press 2. For Stargazer astrophysics news, press 3.

El elemento <enumerate> debe usar el contenido de los prompts y los elementos catch asociados a los elementos <menu> con los elementos <field> que contienen los elementos <option>. Un error semántico debe lanzar el evento *error.semantic* siempre y cuando se está usando en algún sitio el elemento <enumerate>.

2.2.5. Gramáticas

2.2.5.1. Gramáticas de voz

El elemento <grammar> se utiliza para proveer una gramática de voz que debe:

- Especificar un conjunto de palabras que el usuario debe decir para indicar una acción o suministrar información.

- Para una declaración coincidente, devolver la correspondiente interpretación semántica. Esto debe ser un valor simple, una pareja de valores u objeto complejo.

El elemento <grammar> es asignado para soportar cualquier tipo de formato que cumpla los dos siguientes requerimientos:

1) El W3C define el Speech Recognition Grammar Specification (SRGS) [SGRS] como estándar para el formato de las gramáticas que deben admitir todos los intérpretes de VoiceXML.

2) El formato en el que se encuentren definidas las gramáticas será específico de cada plataforma. Las plataformas VoiceXML deben admitir el formato XML Form de la W3C SRGS y deberían soportar el Augmented BNF (ABNF) Form de la W3C SRGS.

En la siguiente tabla se definen los elementos pertenecientes al estándar de *XML form W3C Speech Recognition Grammar.*

Elemento	**Funcionalidad**
<grammar>	Elemento raíz de una gramática XML.
<meta>	Declaración del encabezado del contenido meta de un HTTP equivalente.
<metadata>	Declaración del encabezado del contenido de metadatos XML.
<lexicon>	Declaración del encabezado de un léxico de pronunciación.
<rule>	Declara una expansión de regla con nombre para una gramática.
<token>	Define una palabra u otra entidad que puede servir de entrada.
<ruleref>	Hace referencia a una regla definida localmente o externamente.
<item>	Define una expansión con repetición y probabilidad opcional.
< one-of >	Define un conjunto de expansiones de reglas alternativas.
<example>	Elemento contenido dentro de una definición de la regla que proporciona un ejemplo de entrada que coincide con la regla.
<tag>	Define una cadena arbitraria que es incluida en línea en una expansión que puede utilizarse para la interpretación semántica.

TABLA 2.8. ELEMENTOS DEL XML FORM DE W3C

Se pueden definir las gramáticas de dos formas totalmente diferenciadas. Además, ésta característica es excluyente, no se puede generar una gramática que tenga las dos siguientes características.

- **Gramáticas en líneas.** Las gramáticas en línea se encuentran definidas o especificadas en el contenido de la etiqueta <grammar>. Estas gramáticas deben estar definidas al completo. Es necesario para este tipo de gramáticas englobar todo el contenido de la gramática en una sección CDATA. Para este tipo de gramáticas, el tipo *parameter* especifica un tipo de medio que rige la interpretación del contenido de la gramática.

- **Gramáticas externas.** Las gramáticas externas son las que están definidas por un elemento del formulario. Si el atributo src está definido y dentro de él se está asignando una gramática en línea, se lanzará el evento de error *error.badfetch*.

Cabe destacar en las gramáticas el elemento *weight.* Este elemento define el peso propiamente dicho de la gramática. El peso es un valor positivo sin exponenciales. Si definimos un peso como 1.0, es equivalente a no dar ningún peso a la gramática. Un peso mayor que 1.0, afecta positivamente a la gramática, un peso negativo, afectará negativamente a la misma.

Acertar con el peso apropiado de la gramática es muy complicado, además no siempre afecta a mejorar el reconocimiento de voz del usuario. Las gramáticas DTMF no se ven afectadas por este campo.

Atributos	**Descripción**
src	URI que especifica la ubicación de la gramática y, opcionalmente, si la gramática es externa, un rulename.
scope	Si es "document", la gramática está activa en todos los diálogos del documento actual. Si es "dialog", la gramática está activa a lo largo del formulario actual. Si se omite, la determinación del alcance de la gramática se resuelve mirando el elemento primario.

Atributos	Descripción
type	Se puede especificar el tipo de medio de comunicación mediante el atributo type. Los tipos de medios de comunicación tentativos para el formato de gramática W3C son "aplicación/srgs + xml" para el formato XML y "aplicación/srgs" para gramáticas ABNF.
weight	Especifica el peso de la gramática.

TABLA 2.9. ATRIBUTOS DEL ELEMENTO <GRAMMAR>

2.2.5.2. Gramaticas DTMF

Una gramática DTMF se distingue de una gramática de voz por el atributo “mode” del elemento <grammar>. Las gramáticas DTMF y las gramáticas de voz pueden ser gramáticas de línea o externas y los atributos “type” y “scope” se utilizan de igual forma para ambas. El elemento <grammar> puede utilizarse para definir una gramática DTMF que:

- Especifica el conjunto de pulsaciones de teclas que un usuario puede utilizar para realizar una acción o facilitar información.
- Si coincide con la entrada DTMF, devuelve la interpretación semántica correspondiente.

Para trabajar con este tipo de gramáticas se necesita que las plataformas VoiceXML admitan el formato XML de gramática DTMF.

2.2.6. Salidas del sistema

2.2.6.1. Prompts

El elemento <prompt> controla la salida de audio sintetizada y los archivos de voz. Conceptualmente, prompts es una cola instantánea para emitir sonidos, por lo tanto, la interpretación precede hasta que el usuario necesita proveer al sistema de una entrada. En este punto, los prompts suenan y el sistema espera a la entrada del usuario. Una vez que la entrada se produce, el subsistema de reconocimiento de voz comienza a interpretar.

Éstos son los atributos de los que se compone el elemento <prompt>:

Atributos	Descripción
bargein	Controla si un usuario puede interrumpir un prompt.
bargeintype	Establece el tipo de bargein: 'discurso' o 'hotword'. Se explicará detalladamente en el apartado 4.5.
cond	Expresión que se debe evaluar como „true“ después de su conversión a boolean para que el prompt se reproduzca. El valor por defecto es „true’.
count	Número que permite emitir diferentes prompts si el usuario está haciendo algo repetidamente. Si se omite, por defecto, es "1".
timeout	Tiempo de espera que se utilizará para la siguiente entrada del usuario.
XML: lang	Identificador del idioma para el prompt. Si se omite, se utiliza el valor especificado en el atributo de "XML: lang" del documento.
XML: base	Declara el URI base desde el que se resuelven los URIs relativos en el prompt.

TABLA 2.10. ATRIBUTOS DEL ELEMENTO <PROMPT>

El contenido del elemento <prompt> está modelado en la especificación W3C Speech Markup Language 1.0. **[SSML]**. Los siguientes elementos de marcas están definidos en VoiceXML 2.0.

Elemento	Funcionalidad
<audio>	Especifica los archivos de audio y el texto a reproducir.
<break>	Especifica una pausa en la salida de voz.
<desc>	Proporciona en el elemento <audio> una descripción de la fuente de

Elemento	Funcionalidad
	audio que no es voz.
<emphasis>	Indica que el texto adjunto debe ser hablado con énfasis.
<lexicon>	Especifica un léxico de pronunciación para el *prompt*.
<meta>	Especifica las propiedades meta y "http-equiv" para el *prompt*.
<metadata>	Especifica el contenido de metadatos XML para el *prompt*.
<p>	Identifica el texto adjunto como un párrafo que contiene cero o más sentencias
<phoneme>	Especifica una pronunciación fonética para el texto.
<audio>	Especifica los archivos de audio y el texto a reproducir.
<break>	Especifica una pausa en la salida de voz.
<desc>	Proporciona en el elemento <audio> una descripción de la fuente de audio que no es voz.
<emphasis>	Indica que el texto adjunto debe ser hablado con énfasis.
<lexicon>	Especifica un léxico de pronunciación para el *prompt*.
<meta>	Especifica las propiedades meta y "http-equiv" para el *prompt*.

TABLA 2.11 ELEMENTOS DE LAS MARCAS DE VOZ

Cuando los <prompt> son usados en VoiceXML, tienen unas propiedades adicionales definidas para el <audio>. VoiceXML también permite que los elementos que pertenecen a <prompt> contengan las definiciones de <enumerate> y <value>.

Cuando se requiere que la plataforma procese documentos con uno o más "xml:lang" atributos definidos, ésto no quiere decir que la plataforma tenga que ser multilenguaje. Cuando un lenguaje definido no esté soportado por la plataforma, el evento *error.unsupported.languaje* debe ser lanzado.

2.2.6.2. Prompts de audio

El elemento <prompt> puede consistir en una combinación de archivos de voz grabados o de voz sintetizada.

```
<?xml version="1.0"?>
< vxml version="2.0">
  < form id="ejemploAudio">
    < block>
        <audio src="ficheroGrabado.wav">
    < /block>
  < /form>
< /vxml>
```

FIGURA 2.9 ARCHIVO DE VOZ REPRODUCIDO

Los audios también pueden ser tocados en un <prompt>. Además, podemos dar la dirección de un fichero de voz vía URL. En VoiceXML, también se puede tener una variable de audio previamente grabada.

Los ficheros de audio no sonarán si se da una URL inválida o están en un formato no reconocido por el sistema. El contenido del fichero puede contener texto u otro elemento de audio. Si el fichero de audio no puede ser tocado o el contenido del elemento audio está vacío, nada sonará y no se enviará ningún evento de error. Si no se completa ninguno de los atributos "src" o "expr", el sistema lanzará un evento de error de tipo error.badfetch

2.2.6.3. Elemento <value>

El elemento <value> es utilizado para insertar el valor de una expresión dentro del <prompt>. En la siguiente figura se muestra un ejemplo donde se puede ver cómo se usa un atributo expr.

```
<prompt>
   El nombre introducido es <value expr="nombre"/>
</prompt>
```

FIGURA 2.10 EJEMPLO DE ATRIBUTO <VALUE>

Este ejemplo primero leería los literales que están delante del atributo value y después pronunciaría el valor de la variable nombre. Tambien se puede definir el formato de las variables que se muestran en el campo value, es decir, podemos decirle al sistema cómo debe pronunciar los atributos guardados.

En el siguiente ejemplo, se puede ver cómo el sistema pronuncia una variable de tipo fecha, en este caso el sistema devolvería el valor dicho con un formato fecha.

```
<var name="fecha" expr="'2015/7/27'"/>
<prompt>
   <say-as interpret-as="date">
      <value expr="date"/>
   </say-as>
</prompt>
```

FIGURA 2.11 EJEMPLO DE ATRIBUTO <VALUE>

2.2.6.4. Elemento *bargein*

Si la implementación de la plataforma soporta bargein, el autor de la aplicación puede especificar que un usuario pueda interrumpir un promp usando una entrada DTMF. Esto hace la conversación más rápida, pero no siempre es deseado. Si el autor de la aplicación requiere que un usuario tenga que escuchar todo el contenido de una advertencia, noticia legal, bargein puede ser deshabilitado.

Los usuarios pueden interrumpir cualquier prompt que tenga el atributo bargein a true, pero, por otro lado, deben esperar a que el prompt termine si el atributo está asignado a false. En el caso de que tengamos varios elementos prompt encolados, se mira el atributo de cada uno de los elementos mientras éstos están siendo escuchados. Si el bargein ocurre durante la secuencia, todos los elementos prompts dejaran de sonar.

Cuando el atributo bargein es falso, mientras el prompt está sonando no se permite ninguna entrada DTMF ni ninguna entrada sonora.

A continuación se muestra una tabla con los atributos que componen el elemento bargein.

Atributos	Descripción
speech	Se detendrá el prompt tan pronto como se detecte voz o entrada DTMF.
hotword	El prompt no se detendrá hasta que se detecte una coincidencia completa de una gramática activa.

TABLA 2.12 ATRIBUTOS DEL ELEMENTOS BARGEIN

Cuando el bargein está activo, los atributos pueden ser usados para definir el tipo de bargein que se puede utilizar para cortar el mensaje de voz.

2.2.6.5. Selección de los prompts

Los tapered prompts son aquellos que pueden cambiar con cada intento. De esta forma, los mensajes de solicitud de información pueden volverse más breves en el supuesto que el usuario se vaya familiarizando con la tarea. Los mensajes de ayuda se pueden detallar si el usuario necesita más ayuda, o bien los prompts pueden cambiar con la intención de hacer la interacción más interesante.

El mecanismo de funcionamiento de los tapered prompts es el siguiente: cada ítem de entrada, initial y menú, tienen un contador de prompts interno que se restablece a uno cada vez que se entra en el formulario o en el menú. Cuando el sistema selecciona uno de estos ítems de entrada, el prompt asociado a este ítem incrementará su contador.

2.2.6.6. Atributo *timeout*

El atributo timeout especifica el intervalo de tiempo en silencio permitido, cuando el sistema está esperando una entrada del usuario después de la reproducción del prompt. Si el intervalo de tiempo es superado, la plataforma lanzará un evento noinput.

La razón para permitir los timeouts es soportar los prompt escalados. Por ejemplo, el usuario puede dar cinco segundos para la primera entrada del sistema, y diez segundos para la siguiente.

```
<prompt count="1"  timeout="5s">
    Digame un color.
</prompt>
<prompt count="2" timeout="10s">
    Por favor digame otro color.
</prompt>
```

FIGURA 2.12 EJEMPLO TIMEOUTS

2.2.7. Flujo de control y *scripts*

2.2.7.1. Variables y expresiones

Las variables en VoiceXML son para todos los aspectos equivalentes a las variables de ECMAScript **[ECMASCRIPT]**. Las variables de VoiceXML pueden ser usadas en <script>. También una variable definida en <script> puede ser usada en VoiceXML. Declarar una variable usando <var> es equivalente a usar la instrucción 'var' en un elemento <script>. Tanto los elementos <var> como <script> solo pueden ser declarados dentro de la etiqueda form.

Las variables son declaradas por el elemento <var>:

<var name="teléfono_casa"/>

<var name="pi" expr="3.14159"/>

<var name="ciudad" expr="'Madrid"/>

Solo pueden ser declarados por el ítem form.

<field name="num_tickets">

  <grammar type="application/srgs+xml" src="/grammars/number.grxml"/>

  <prompt>¿Cuantos tiques deseas comprar?</prompt>

</field>

Las variables que no son declaradas con un valor inicial son inicializadas con el valor ECMAScript sin definir. Las variables tienen que ser declaradas antes de ser usadas en VoiceXML o ECMAScript. Si usamos una variable sin declarar en ECMAScript el error error.semantic será lanzado por el sistema.

En un formulario, las variables declaradas por <var> y esas variables declaradas por los ítems del formulario, son inicializadas cuando el formulario es cargado. En la siguiente imagen se puede apreciar la declaración de variables de los modos explicados anteriormente.

```
<form id="test">
  <var name="uno" expr="1"/>
    <field name="dos" expr="uno+1">
      <grammar type="application/srgs+xml" src="/grammars/number.grxml"/>
    </field>
    <var name="tres" expr="two+1"/>
    <field name="go_on" type="boolean">
      <prompt>Diga si o no para continuar.</prompt>
    </field>
</form>
```

FIGURA 2.13 EJEMPLO DECLARACIÓN DE VARIABLES

Como se puede observar, cuando el usuario visita el formulario, la inicialización del formulario declara la variable uno y le asigna el valor de uno. Después declara la variable dos y le da el valor dos. A continuación, declara la variable tres y le da el valor tres. Por último, el algoritmo de interpretación del formulario interpreta que debe comenzar con el campo go_on.

2.2.7.1.1. Alcance de las variables

VoiceXML usa el alcance de ECMAScript para permitir a las variables ser declaradas en diferentes niveles en la jerarquía de la aplicación. Por ejemplo, una variable declarada en el ámbito del documento puede ser referenciada desde cualquier sitio del documento. Las variables pueden definirse desde los siguientes ámbitos.

Ámbito	Descripción
sesión	Variables de sólo lectura que pertenecen a toda una sesión de usuario. Son declaradas por el intérprete de contexto.
aplicación	Variables que se declaran con elementos <var> y <script> como hijas del documento raíz <vxml>. Se inicializan cuando se carga el documento raíz de la aplicación. Existen mientras el documento raíz está cargado y son visibles para cualquier documento de la aplicación.
documento	Variables que se declaran con <var> y <script> en un documento concreto que no sea el documento raíz. Se inicializan cuando se carga

Ámbito	Descripción
	el documento y son accesibles sólo dentro de ese documento.
diálogo	Variables declaradas como hijas de <form> o <menu>. Son accesibles sólo dentro de ese elemento de diálogo.
anónimo	Cada elemento <block>, <filled>, y <catch> definen un nuevo alcance anónimo en el que se pueden

TABLA 2.13 ÁMBITO DE LAS VARIABLES

A continuación se presenta un diagrama que muestra la herencia de los ámbitos.

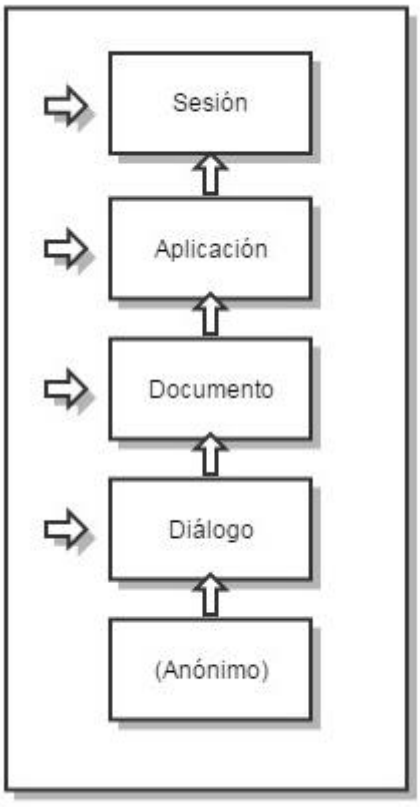

FIGURA 2.14 DIAGRAMA ÁMBITO DE LAS VARIABLES

No es nada recomendable usar “sesión”, “aplicación”, “documento”, y “diálogo” como nombres de variable e ítems de formulario. Aunque no son palabras reservadas, al utilizarlas se ocultan las variables predefinidas con el mismo nombre por las reglas de ámbito de ECMAScript utilizadas por VoiceXML.

2.2.7.1.2. Variables referenciadas

Las variables son referenciadas por los atributos *cond* y *expr*: Seguidamente, se muestra un ejemplo de cómo referenciar variables.

```
<if cond="ciudad == 'Leganés'">
    <assign name="ciudad" expr="'Madrid'"/>
<elseif cond="city == 'Alcorcón'"/>
    <assign name="ciudad" expr="'Madrid'"/>
<elseif cond="ciudad =='Illescas'"/>
    <assign name="ciudad" expr="'Toledo'"/>
</if>
```

FIGURA 2.15 EJEMPLO DE REFERENCIA DE VARIABLES

2.2.7.2. Manejo de eventos

La plataforma lanza eventos cuando el usuario no responde, o no responde en un camino que la aplicación sea capaz de entender, como pedir ayuda. El intérprete lanza eventos si encuentra un error semántico en el documento VoiceXML, o cuando encuentra el elemento <throw>. Los eventos son identificados por un string de caracteres.

Cada elemento, cuando es lanzado, tiene una serie de manejadores concretos. A continuación se puede observar una lista de los manejadores de eventos.

- <catch>
- <error>
- <help>
- <noinput>
- <nomatch>

El elemento hereda los catch elements de cada uno de los elementos antecesores. Si un campo, por ejemplo, no contiene un catch element por no coincidencia, pero el formulario sí lo tiene, el nomatch del formulario puede capturar este evento en cuestión. Siguiendo este camino, un manejador de un evento común puede ser especificado a cualquier nivel, y ser aplicado a todos sus niveles descendientes.

2.2.7.2.1. Elemento throw

El elemento <throw> es utilizado para lanzar eventos. Dos ejemplos de cómo lanzar eventos son los siguientes:

```
<throw event="nomatch"/>
<throw event="connection.disconnect.hangup"/>
```

FIGURA 2.16 EJEMPLO DE LANZAMIENTO DE EVENTOS

A continuación, se muestra el ejemplo de un evento de aplicación lanzado.

```
<throw event="com.att.portal.machine"/>
```

FIGURA 2.17 EJEMPLO LANZAR EVENTO DE APLICACIÓN

Los atributos de <throw> son los siguientes.

Atributos	Descripción
event	Evento que se produce.
eventexpr	Expresión ECMAScript que evalúa el nombre del evento que se produce.
message	Cadena de mensaje que proporciona contexto adicional sobre el evento que se produce.
messageexpr	Expresión ECMAScript que evalúa la cadena del mensaje.

TABLA 2.14 ATRIBUTOS DEL ELEMENTO THROW

Se debe especificar uno de los atributos "*event*" o "*eventexpr*"; de lo contrario, se produce un evento error.badfetch.

2.2.7.2.2. Elemento catch

El elemento catch es el elemento encargado de capturar eventos en un documento, diálogo, o formulario (excepto en blocks). Éstos contienen contenido ejecutable.

El elemento catch anónimo contiene la variable _event, la cual contiene el nombre del evento que ha sido lanzado. Por ejemplo, en la siguiente imagen, se observa cómo se

evalúa la variable _event para mostrar diferentes tipos de archivo .wav, dependiendo del valor de la misma.

```
<catch event="event.foo event.bar">
  <if cond="_event=='event.foo'">
    <audio src="foo.wav"/>
  <else/>
    <audio src="bar.wav"/>
  </if>
</catch>
```

FIGURA 2.18 EJEMPLO CAPTURA DE EVENTO, EVALUANDO EL TIPO DE EVENTO

Se debe tener sumo cuidado a la hora de capturar eventos para volver a lanzar uno nuevo después, si lanzamos el mismo evento que hemos capturado, podríamos formar un bucle infinito. La plataforma podría detectar esta situación y lanzar un evento de error semántico.

En la siguiente tabla se observan los atributos de elemento <catch>.

Atributos	Descripción
event	Evento o eventos a capturar. Puede especificarse una lista de eventos separados por espacios, que indicará que el elemento <*catch*> detecta todos los eventos de la lista. En este caso existe un contador para cada evento. Si no se especifica el atributo, todos los eventos deben ser capturados.
count	Contador de las apariciones del evento. El valor por defecto es 1. El recuento permite controlar las veces que aparece el mismo evento.
cond	Expresión que debe ser *true* después de su conversión a

TABLA 2.15 ATRIBUTOS DEL ELEMENTO CATCH

2.2.7.2.3. Nota abreviada.

Los elementos <error>, <help> y <nomatch> son abreviaturas para los tipos más comunes de los elementos catch.

- <error>: Es la abreviatura de <catch event="error"> y sirve para capturar todos los eventos relacionados con errores.

- <help>: Es el elemento abreviado de <catch event="help"> y sirve para asistir al usuario cuando menciona la palabra ayuda.

- <noinput>: Es el elemento abreviado de <catch event="noinput">, es lanzado cuando no se ha recibido ningún tipo de sonido por parte del usuario.

- <nomatch>: Es el elemento abreviado de <catch event="nomactch">, es lanzado cuando lo que dice el usuario no tiene ninguna coincidencia con la gramática.

Atributos	Descripción
count	Es el contador del evento.
cond	Es una condición opcional, para testear si el evento es capturado por este elemento. Por defecto está a true.

TABLA 2.16 ATRIBUTOS DE LOS ELEMENTOS DEFINIDOS

2.2.7.2.4. Elementos catch por defecto

El intérprete tiene definidos o proporciona una serie de manejadores de elementos catch por los siguientes tipos de eventos: noinput, help, nomatch, cancel, exit y eventos de error si el autor no especifica nada.

Los manejadores de los diferentes eventos están resumidos en la siguiente tabla que se muestra a continuación. En ella se puede ver cuándo un audio está configurado para que responda y cómo afecta este tipo de eventos a la ejecución.

Tipo de evento	Audio	Acción
Cancel	**No**	No repetición.
Error	**Si**	Salida del intérprete.
Exit	**No**	Salida del intérprete.
Help	**Si**	Repetición

Tipo de evento	Audio	Acción
Noinput	**No**	Repetición
Nomatch	**Si**	Repetición
Maxspeechtimeout	**Si**	Repetición
Connection.disconnect	**No**	Salida del intérprete.
cond	**Si**	Salida del intérprete.

TABLA 2.17 MANEJADORES CATCH POR DEFECTO

La especificación de las plataformas puede ser diferente en los prompts por defecto presentados.

2.2.7.2.5. Tipos de eventos

Podemos diferenciar diferentes tipos de eventos: los predefinidos, los de aplicación y los específicos de la plataforma. Los eventos están subdivididos en eventos sencillos, que son cosas que ocurren con normalidad, o eventos de error, éstos son cosas que ocurren de forma anormal.

Los eventos predefinidos son los siguientes.

- Cancel: el usuario realiza una petición de cancelación.
- Connection.desconnect.hangup: El usuario ha realizado un “hung up”.
- Connection.desconnect.transfer: El usuario ha sido transferido incondicionalmente a otra línea y no puede volver.
- Exit: El usuario ha realizado una petición de salida.
- Help: El usuario ha realizado una petición de ayuda.
- Noinput: El usuario no ha respondido en el intervalo de tiempo correspondiente.
- Nomatch: El usuario ha realizado una entrada, pero no ha sido reconocida por el sistema.

- Maxspeechtimeout: La entrada del usuario ha durado demasiado tiempo, se ha excedido del tiempo determinado previamente.

Además de errores de transferencia, los errores predefinidos son:

- Error.badfetch: El contexto del intérprete lanza este evento cuando se produce un error en la captura de un documento y se llega a un lugar donde se requiere ese resultado.
- Error.badfetch.http.response_code
- Error.badfetch.protocol.response_code: En el caso de un fallo de fetch, el contexto del intérprete debe utilizar un evento detallado diciendo qué código de respuesta se ha producido.
- Error.semantic: Se encontró un error de tiempo de ejecución en el documento de VoiceXML.
- Error.noauthorization: Se produce cuando la aplicación intenta realizar una operación que no está autorizada por la plataforma.
- Error.noresource: Se ha producido un error en tiempo de ejecución debido a que un recurso de plataforma solicitado no estaba disponible durante la ejecución.
- Error.unsupported.builtin: La plataforma no admite un tipo de gramática solicitado.
- Error.unsupported.format: El recurso solicitado tiene un formato que no es compatible con la plataforma.
- Error.unsupported.language: La plataforma no admite el lenguaje para síntesis o reconocimiento de voz.
- Error.unsupported.objectname: La plataforma no admite un determinado objeto específico de la plataforma.
- Error.unsupported.element: La plataforma no admite un determinado elemento de VoiceXML.

2.2.7.3. Contenidos ejecutables

Contenidos ejecutables se refiere a un bloque o crecimiento lógico, como puede ser un:

- <Block>.
- <Filled>.
- Manejadores de eventos.

Los elementos ejecutables son ejecutados en un documento ordenado por un bloque de procedimiento lógico. Si un elemento ejecutable provoca un error, ese error es lanzado inmediatamente. Los ejecutables siguientes en ese bloque de procedimiento lógico no son ejecutados.

2.2.7.3.1. Elemento var

Este elemento declara una variable. Esto puede ocurrir en un contenido ejecutable o como un hijo de <form> o <vxml>.

Si se declara en un contenido ejecutable, esta declaración es únicamente realizada cuando el elemento <var> es ejecutado. Si la variable ya ha sido declarada en el ámbito del documento, las declaraciones posteriores solo reasignan el nuevo valor a la variable.

Si está declarada como un hijo del elemento <form>, se declara una variable en el ámbito del diálogo del <form>. Esta declaración se realiza durante la inicialización del formulario. Esta variable no es un ítem del formulario por lo que no será visitada por el algoritmo de interpretación del formulario.

Si por el contrario está declarada como un hijo del elemento <vxml>, esta variable será declarada en el ámbito del documento. Si cualquier hijo del documento desea utilízala, solo debe declararla en el ámbito de la aplicación. Las inicializaciones ocurren en el orden de los documentos, en las inicializaciones de los mismos.

Mostramos una tabla con los atributos del elemento <var>

Atributo	Descripción
name	El nombre de la variable que va a obtener el resultado. Este atributo no puede especificar una variable con el ámbito de un prefijo, si eso ocurre será lanzado un evento de error.semantic.
expr	Es el valor inicial de la variable. Este atributo es opcional, si no aparece, la variable retiene su valor actual.

TABLA 2.18 ATRIBUTOS DEL ELEMENTO <VAR>

2.2.7.3.2. Elemento assign

El elemento <assign>, es el encargado de asignar un valor a una variable.

Es ilegal realizar una asignación de una variable que no ha sido explícitamente declarada usando el elemento <var>. Si realizamos dicha asignación, el sistema lanzará un evento de error.semantic.

Mostramos una tabla con los atributos de este elemento.

Atributo	Descripción

Atributo	Descripción
nombre	El nombre de la variable que va a ser asignada. Por defecto, el ámbito de la variable es el más cercano al ámbito del elemento activo.
expr	El nuevo valor de la variable.

TABLA 2.19 ATRIBUTOS DEL ELEMENTO <ASSIGN>

2.2.7.3.3. Elemento clear

El elemento <clear> resetea el valor de una o más variables, incluidos los ítems del formulario. Una vez que una variable declarada ha sido identificada, su valor es asignado a ECMAScript undefined value. Además si el nombre de la variable corresponde con el ítem del formulario, los ítems del formulario (prompt counters y event counters) son reseteados.

Atributo	Descripción
namelist	Lista de variables cuyos valores se van a restablecer. Cuando este atributo no se especifica, se reinician todos los ítems de formulario en el formulario actual.

TABLA 2.20 ATRIBUTOS DEL ELEMENTO <CLEAR>

2.2.7.3.4. Elementos if, elseif y else

El element if es usado para condiciones lógicas. Este elemento tiene como opcional los otros dos elementos, <else> e <elseif>.

En definitiva, cuando realizamos una operación lógica if, necesitamos un elemento de comparación. Si se cumple este elemento, pasaríamos a ejecutar las líneas de código que están definidas dentro de él. Si por el contrario no se cumple esta condición, pasaríamos a ejecutar las líneas de código que están definidas en el elemento else. Por último, si realmente la condición no se cumple pero queremos que la opción else se ejecute algunas veces, deberemos utilizar el elemento elseif, que agrega una nueva condición de comparación de estudio.

Podemos anidar tantos elseif como queramos, a continuación se muestra un ejemplo de uso de estos elementos:

```
<if cond="total > 1000">
    <prompt>Demasiado dinero gastado.</prompt>
    <throw event="com.xyzcorp.acct.toomuchspent"/>
</if>
<if cond="cantidad &lt; 29.95">
    <assign name="x" expr="cantidad"/>
<else/>
    <assign name="x" expr="29.95"/>
</if>
<if cond="sabor == 'vainilla'">
    <assign name="codigo_sabor" expr="'v'"/>
<elseif cond="sabor == 'chocolate'"/>
    <assign name="codigo_sabor" expr="'h'"/>
<elseif cond="sabor == 'fresa'"/>
    <assign name="codigo_sabor" expr="'b'"/>
<else/>
    <assign name="codigo_sabor" expr="'?'"/>
</if>
```

FIGURA 2.19 EJEMPLO USO ELEMENTOS IF/ELSE/ELSEIF

2.2.7.3.5. Elemento reprompts

La FIA espera un elemento catch para encolar los promps apropiadamente según se esté manejando un evento. La FIA generalmente no realiza la selección y el encolamiento de los prompts en la siguiente iteración tras la ejecución del elemento catch. Sin embargo, la FIA realiza la selección y encolamiento normal de los prompts después de la ejecución de un elemento catch (<catch>, <error>, <help>, <noinput>, <nomatch>) en dos casos:

- Si el elemento catch finaliza por la ejecución de un elemento <goto> o <submit> a otro diálogo, o si también es terminado con el elemento <return> desde un subdiálogo.

- Si un <reprompt> es ejecutado en el catch para solicitar que se reproduzcan los prompts posteriores.

En estos dos casos, después de que la FIA seleccione el siguiente ítem del formulario a visitar, realiza el procesamiento normal de los prompts, incluyendo la selección y el encolamiento de los prompts ítems pertenecientes al formulario e incrementa el contador de los mismos.

2.2.7.3.6. Elemento goto

El element goto tiene los siguientes usos:

- Transición a otro ítem dentro del mismo formulario.

- Transición a otro diálogo en el mismo documento.

- Transición a otro documento.

Para realizar la transición a otro ítem del formulario, usa el siguiente nextitem atributo, o el expritem atributo si el nombre del ítem del formulario se calcula usando la expresión ECMAScript.

```
<goto nextitem="confirmacion"/>
<goto expritem="(type==12)? 'ssn_confirm' : 'anulacion'"/>
```

FIGURA 2.20 EJEMPLO GOTO A OTRO ÍTEM DEL FORMULARIO

Para ir a otro diálogo del mismo documento, usamos next (o expr) con el fragmento de la URI del diálogo.

```
<goto next="#otro_dialogo"/>
<goto expr="'#' + 'otro_dialogo'"/>
```

FIGURA 2.21 EJEMPLO GOTO A OTRO DIÁLOGO

Para realizar la transición a otro documento, es necesario utilizar el elemento next (o expr) con la dirección URI del documento.

```
<goto next="http://flight.example.com/reserve_seat"/>
<goto next="./special_lunch#wants_vegan"/>
```

FIGURA 2.22 EJEMPLO GOTO A OTRO DOCUMENTO

2.2.7.3.7. Elemento submit

El elemento <submit> es usado para enviar información al servidor web origen y al final de la transición del documento mandar de vuelta la respuesta. A diferencia de <goto>, este elemento permite enviar una lista de variables del documento vía HTTP GET o POST **[HTTP]**.

El diálogo al que se transita está especificado por la URI referenciada en el atributo expr o next del elemento submit. La URI es buscada siempre incluso si contiene solo un fragmento. En el caso de los fragmentos, la URI referenciada es la base de la URI del actual documento.

A continuación se definen los atributos del campo submit:

Atributo	Descripción
next	Referencia URI del documento al que se quieren enviar las variables.
expr	Lo mismo que *next*, excepto que la referencia URI será determinada de forma dinámica mediante la evaluación de la expresión dada de ECMAScript.
namelist	Lista de variables a enviar separadas por espacios en blanco. Este atributo es opcional. Si no se especifica, se enviarán todas la variables declaradas mediante el atributo *name* en el campo del formulario que contenga al <submit>.
method	Método de solicitud: *get* (por defecto) o *post*.
enctype	Tipo de medio de codificación del documento enviado (cuando el valor de *method* es "post").

TABLA 2.21 ATRIBUTOS DEL ELEMENTO <SUBMIT>

Es obligatorio especificar uno de los atributos *next* o *expr*, de lo contrario, se lanzará un evento de error *error.badfetch.*

2.2.7.3.8. Elemento exit

Devuelve el control al context, el cual determina qué hacer después.

Este elemento es diferente a <return>, <exit> es el que termina o finaliza todos los documentos cargados, mientras que <return> vuelve desde una invocación de un <subdialog>. Si el subdiálogo provocó un nuevo documento o aplicación para ser invocado, entonces <return> causará que el documento sea terminado, pero la ejecución se reanudará después del <subdialog>.

El elemento <exit> no provoca un evento “exit”.

2.2.7.3.9. Elemento return

El elemento return termina la ejecución de un subdiálogo y devuelve el control y el valor a la llamada del diálogo.

A continuación se muestra una tabla con los atributos del elemento return:

Atributo	Descripción
event	Vuelve y, a continuación, lanza este evento.
eventexpr	Vuelve y, a continuación, inicia el evento al que esta expresión de ECMAScript evalúa.
message	Cadena de mensaje que proporciona contexto adicional sobre el evento que se produce.
messageexpr	Expresión de ECMAScript que evalúa a la cadena de mensaje.
namelist	Nombres de las variables que se devuelven al diálogo que llama. El valor por defecto es no devolver ninguna variable.

TABLA 2.22 ATRIBUTOS DEL ELEMENTO <RETURN>

En el retorno de un subdiálogo, un evento puede ser lanzado en el punto de invocación o cuando un dato es devuelto como un objeto CMAScript con propiedades correspondientes a la variable especificada en su lista de nombres. Un elemento devuelto, que es encontrado cuando no se ejecuta como un diálogo, lanza una excepción de error semántico.

Este elemento debe tener al menos un atributo definido “event”, “eventexpr” o “namelist”, de lo contrario se lanzará una excepción *error.badfetch*.

2.2.7.3.10. Elemento log

El elemento log es una herramienta que puede ayudar a seguir el código a la hora de la depuración. Cuando se define este tipo de elemento, se manda un mensaje al servidor central, dejando un registro del mensaje en el log. Podemos ver todos estos mensajes con la herramienta “application debugger” de la Plataforma Voxeo.

2.3. Plataforma IVR

La respuesta de voz interactiva o IVR (Interactive Voice Response) consiste en un sistema capaz de interactuar con el ser humano. Antes de la evolución de este tipo de tecnologías, el sistema solo era capaz de reconocer respuestas simples, tales como si, no u otras frases concretas y cortas. Estos sistemas utilizaban pre-grabados de voz, dando a

entender menús posibles a utilizar por el usuario, los cuales introducían la respuesta a través de tonos del teclado del teléfono.

Actualmente, se da la posibilidad al usuario tanto de introducir respuestas por teclado, como por voz. El sistema es totalmente capaz de reconocer la voz e interpretar su entrada con el fin adecuado.

Este tipo de tecnología es usada por empresas que reciben enormes cantidades de llamadas al día. De esta manera el número de operadores se ve reducido en un número considerable. Las empresas usan la tecnología IVR para realizar el primer sondeo al usuario, es decir, captan la llamada del usuario, ven cuál es su necesidad y a posteriori, le pone en contacto con el departamento solicitado. De esta manera se consigue que el sistema sea más dinámico, el servicio más rápido y a su vez, se evita al usuario el salto a diferentes agentes especializados.

Para mejorar sus servicios, el IVR involucra otras tecnologías, tales como DTMF (Dual Tone Multi Frequency), TTS (Text To Speech) y ASR (Automatic Speech Recognition).

2.3.1. Plataforma VoiceCenter

La Plataforma VoiceCenter, permite crear, implementar y probar aplicaciones desarrolladas en VoiceXML. Esta plataforma está desarrollada por Voxeo Corporation (*www.voxeo.com*).

Además incluye una web local para permitir el alojamiento de las aplicaciones de voz y un número de teléfono asociado a esta aplicación, para poder usarlo durante la prueba de estas aplicaciones. En esta plataforma también podemos encontrar múltiples enlaces de documentación y un foro para presentar tus dudas que resulta bastante útil a la hora de encontrar soluciones de implementación.

El funcionamiento de Voxeo es muy similar al de un servidor web. En los servidores web, los navegadores realizan peticiones por medio de los protocolos de internet. Los servidores web también alojan las páginas estáticas que son vistas desde el navegador. En voxeo, las aplicaciones tienen las mismas características, pero no son vistas desde el navegador, “se navegan” desde el elemento utilizado para la llamada. Por lo tanto, estas aplicaciones tienen el mismo esquema que cualquier aplicación utilizada desde internet.

2.3.1.1. Primera toma de contacto en la plataforma Voxeo

En este apartado se define qué pasos se deben seguir en la plataforma Voxeo para publicar una aplicación y realizar sus posteriores pruebas de depuración.

El primer paso es crear una cuenta de acceso para poder interactuar en su sistema, para lo cual, se accede a la página www.evolution.voxeo.com y se presiona el botón "create an account".

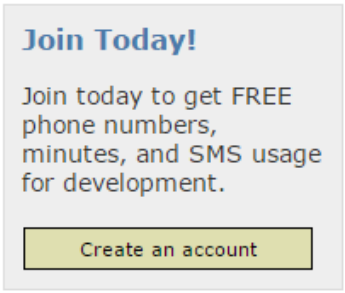

FIGURA 2.23 BOTÓN A PRESIONAR PARA REALIZAR UN ALTA EN EL SISTEMA VOXEO

Posteriormente, se aceptan los términos del servicio, se rellenan los datos de la cuenta y se pasa el Security Test. Una vez completados estos pasos, se podrá realizar el loggin.

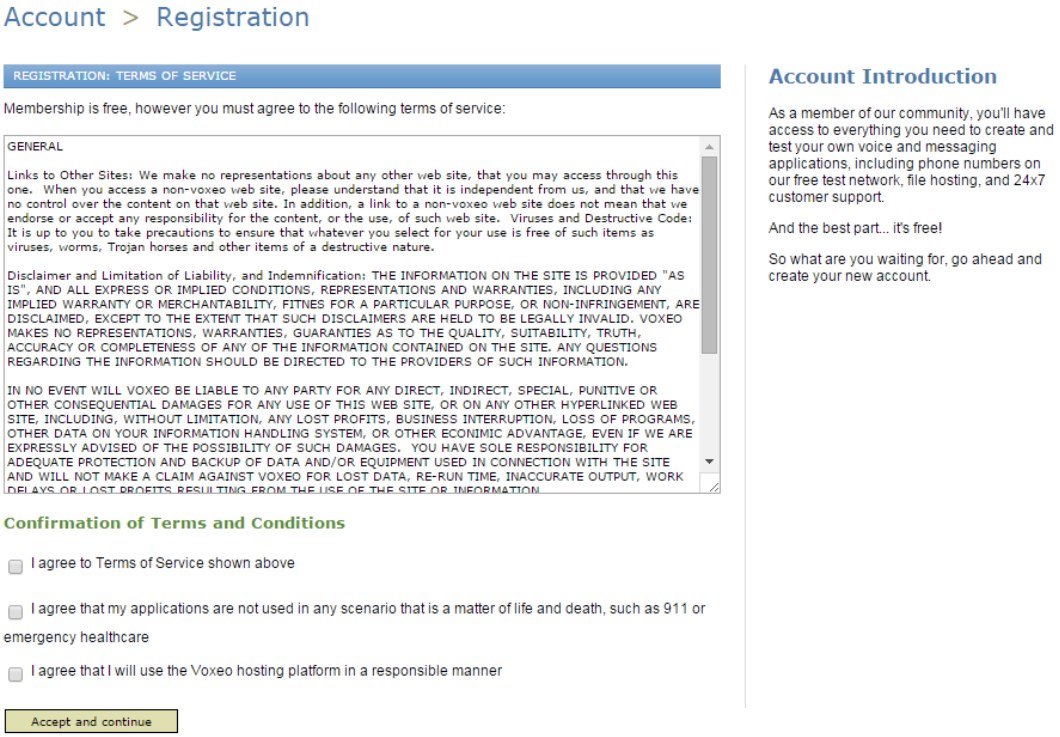

FIGURA 2.24 PÁGINA DE TÉRMINOS DE USO

Una vez realizado el loggin en la plataforma, el sistema muestra una vista general de las herramientas que se pueden utilizar durante la experiencia con el sistema.

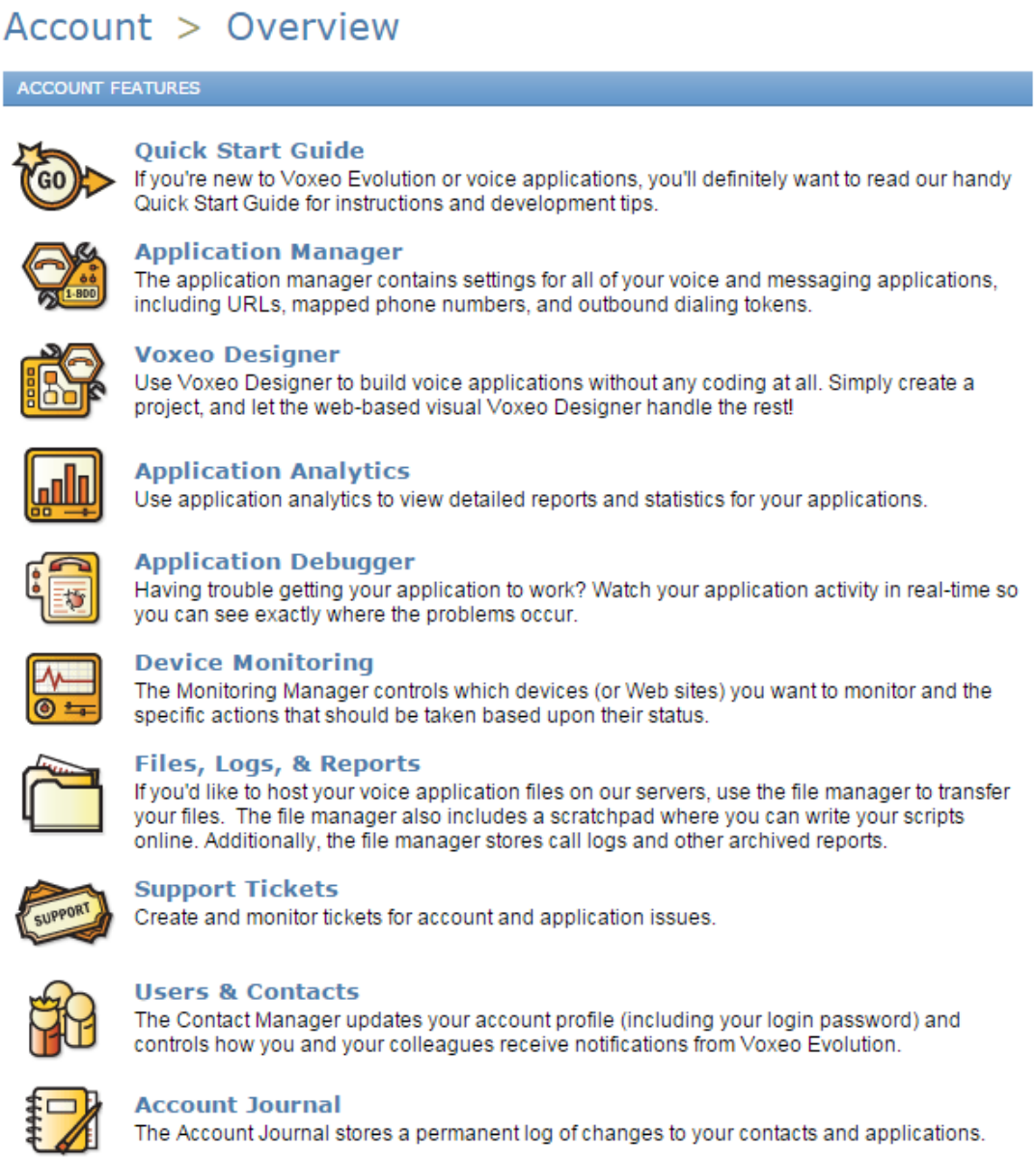

FIGURA 2.25 PÁGINA OVERVIEW DE LA CUENTA DADA DE ALTA

A continuación se pasa a definir las diferentes herramientas que ofrece el sistema:

- **Quick Start Guide:** Como su nombre indica, se trata de una guía rápida donde se explica al usuario los pasos a seguir para empezar a utilizar la plataforma.

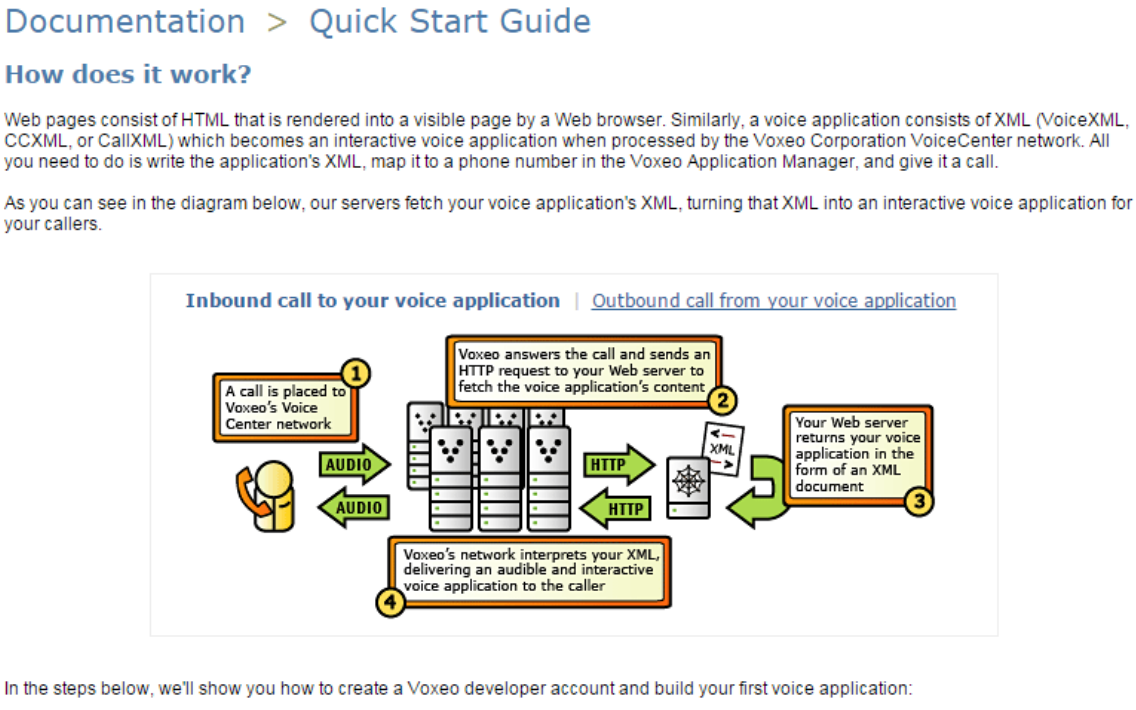

FIGURA 2.26 GUIA RÁPIDA PARA EMPEZAR A UTILIZAR VOXEO

- **Application Manager:** La herramienta Application Manager es la encargada de ofrecer al usuario las diferentes aplicaciones que tiene dadas de alta en el sistema para proceder a su utilización o configuración. También da la opción que crear nuevas aplicaciones.

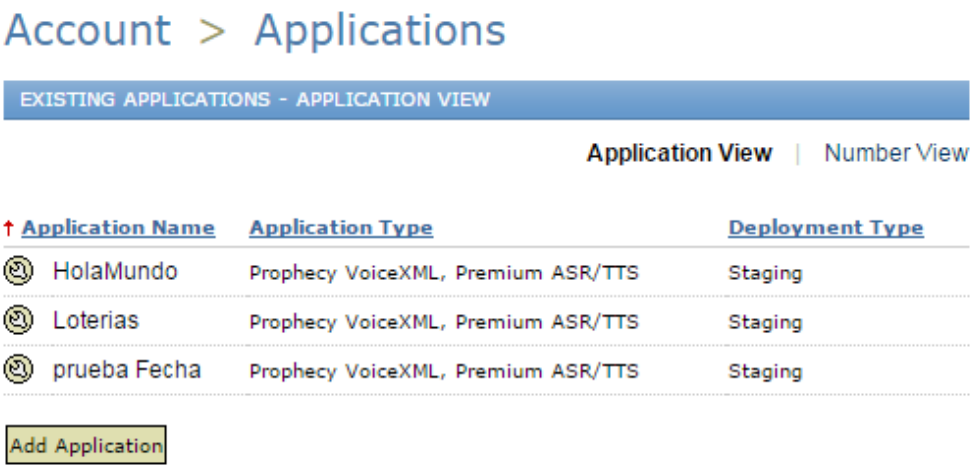

FIGURA 2.27 APPLICATION MANAGER

- **Voxeo Designer:** Se utiliza Voxeo Designer para crear aplicaciones de voz sin necesidad de codificación. Simplemente, se crea un proyecto de forma gráfica y Voxeo Designer lo gestiona.

- **Application Analytics:** Esta herramienta sirve para ver detalles de los reportes y las estadísticas de las aplicaciones que se tienen dadas de alta en el sistema.

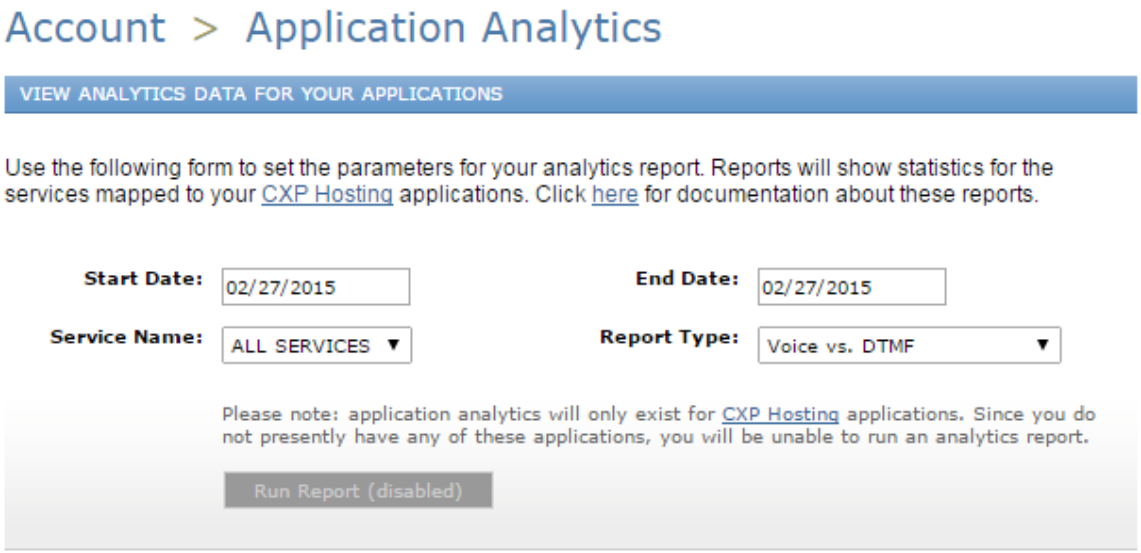

FIGURA 2.28. APPLICATION ANALYTICS

- **Application Debugger:** Esta es la herramienta utilizada para depurar la aplicación en tiempo real. Se puede comprobar la actividad de la aplicación, además de poder ver exactamente dónde está ocurriendo el error de la aplicación publicada.

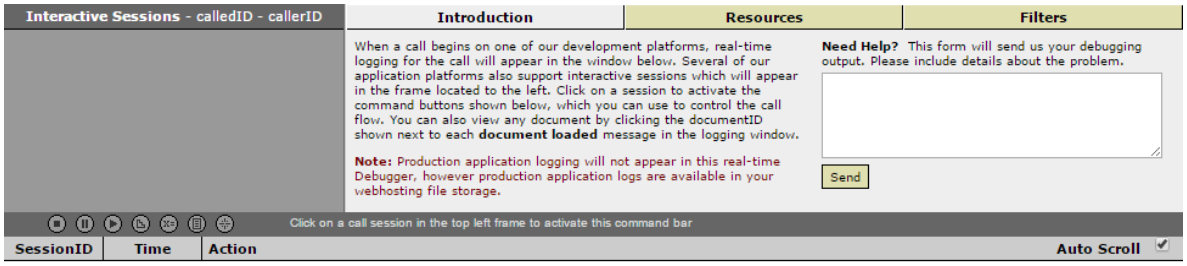

FIGURA 2.29 APPLICATION DEBUGGER

- **Device Monitoring:** La herramienta Device Monitoring se utiliza para controlar qué dispositivos (o sitios web) se quieren monitorizar y qué acciones específicas fueron tomadas basadas en su estado.

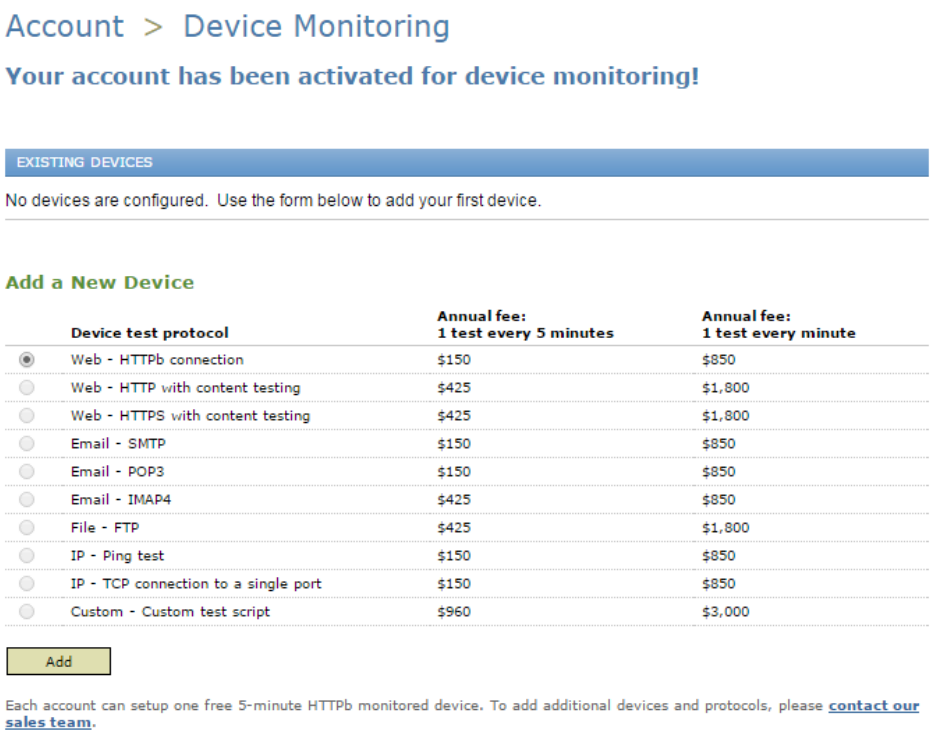

FIGURA 2.30 DEVICE MONITORING

- **Files, Logs, & Reports:** Si el usuario desea alojar los archivos de las aplicaciones Voice en sus servidores, se puede usar esta herramienta para transferir dichos ficheros. También cuenta con un *scratchpad* donde se pueden escribir o editar los archivos online. Adicionalmente, esta herramienta tiene un log de llamadas y reportes archivados.

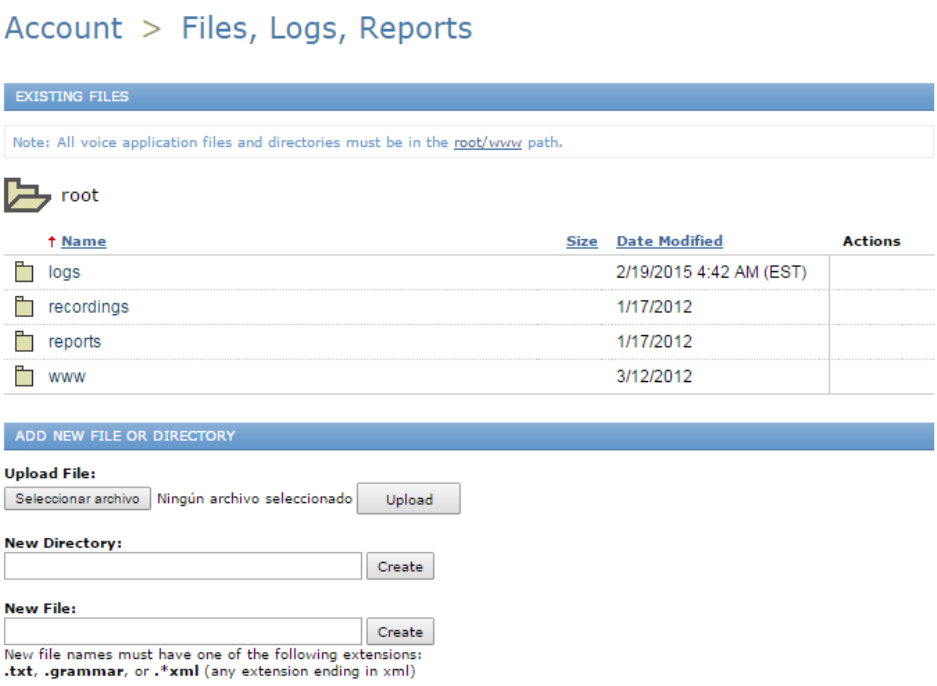

FIGURA 2.31 FILES, LOGS, REPORTS MANAGER

- **Support Tickets:** Con la herramienta Support Tickets se crean y controlan los tickets para cuestiones de cuenta y de aplicación.

- **Users & Contacts:** Esta herramienta actualiza las cuentas de usuario (incluido el loggin y el password), controla cuántas notificaciones reciben sus compañeros o las propias aplicaciones desde Voxeo evolution.

- **Account Journal:** La herramienta Account Journal guarda los logs de carga de los contactos y aplicaciones.

2.3.1.2. Aplicaciones en VoiceXML

Se trata de un sistema muy intuitivo en el que fácilmente se puede alojar una aplicación.

A continuación se muestran los pasos a seguir si se desea dar de alta una aplicación:

El primer menú a seleccionar es Application Manager, en el que, tal y como se ha explicado anteriormente, se ve una lista de aplicaciones. En este caso está vacía. Para añadir una aplicación, se debe pulsar el botón "Add Applicatión".

Una vez presionado este botón, aparece la siguiente pantalla:

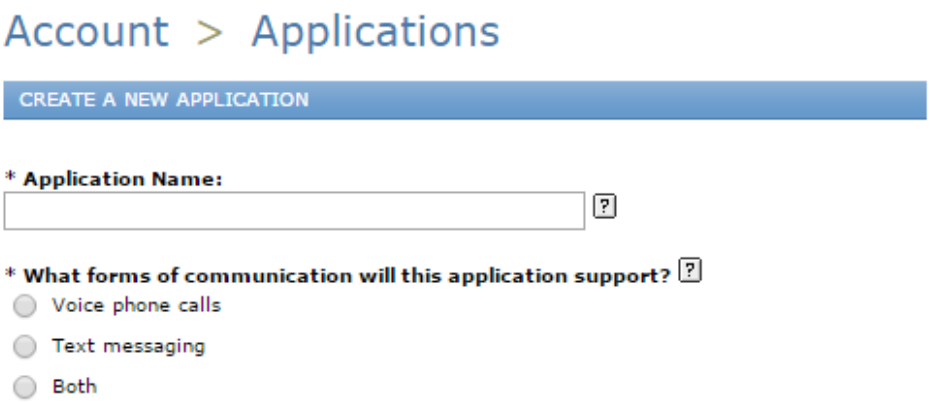

FIGURA 2.32 CREAR UNA NUEVA APLICACIÓN

Para crear una aplicación, previamente es necesario seleccionar la forma de comunicación que ésta va a soportar. Existen tres tipos diferentes:

- Llamadas telefónicas: Esta aplicación se encargará de responder a llamadas telefónicas únicamente.

Cuando se selecciona esta opción, aparece un cuadro donde se pueden configurar algunas opciones. Se puede observar que se van ofreciendo más opciones de configuración según se van seleccionando los cuadros anteriores.

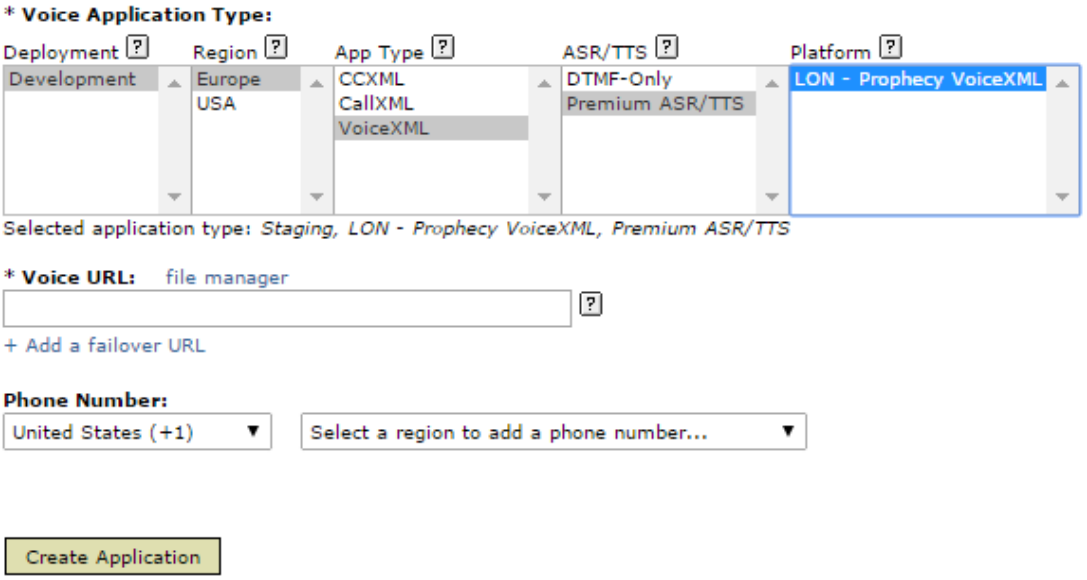

FIGURA 2.33 VOICE APPLICATION TYPE

Cabe destacar uno de los text box de selección, *App Type*. En este cuadro se selecciona la plataforma de la aplicación de voz que se va a utilizar. Se pueden seleccionar tres tipos diferentes de plataformas:

- CCXML está diseñado para proveer un soporte de control de llamadas telefónicas al igual que VoiceXML. Esta plataforma es una buena elección si se quieren hospedar aplicaciones de conferencia de última generación.

- CallXML 3.0 está disponible únicamente en Voxeo. Conserva su bajo coste y corta curva de aprendizaje, la versión 3.0 añade la habilidad de usar menús simples de reconocimiento de voz y emplea estructuras de operaciones de lógica condicional para extender sus capacidades.

  Si la aplicación no va a necesitar reconocimientos de voz complejos, CallXML es la opción más recomendada a elegir.

- VoiceXML 2.1 es la plataforma que soporta todas las mejoras, siempre que *SISR/SRGS grammar* siga formateando los estándares. También incluye el soporte para los formatos de gramática antiguos, ofreciendo una gran flexibilidad y variedad de opciones a la hora de implementar aplicaciones de voz.

- Mensajes de texto: La aplicación desarrollada para este proyecto permite recibir y contestar mensajes mediante mensajes de texto.

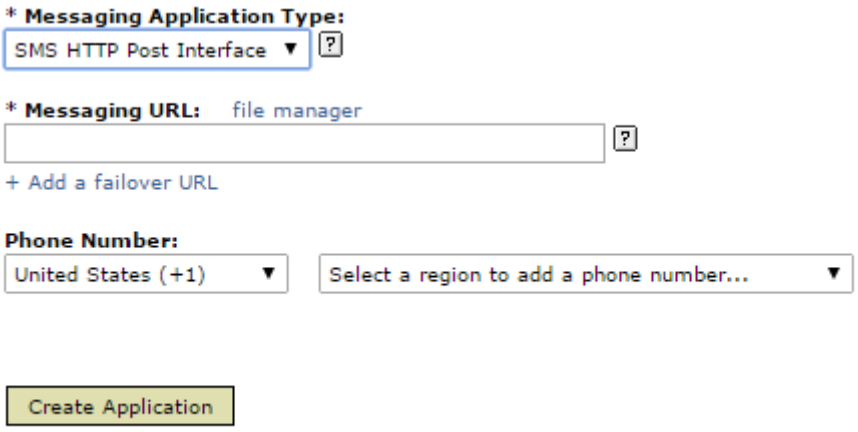

FIGURA 2.34 MESSAGING APPLICATION TYPE

En este caso solo se dispone de dos tipos de aplicaciones.

- CXP 14 w/ SMS.
- SMS HTTP Post Interfaze.
- Ambos: En este caso esta aplicación soporta las dos anteriores.

Cuando todas las opciones están completadas, será necesario aportar la URL de la aplicación y pulsar "Create Application".

A continuación, aparecerá un resumen de la aplicación generada.

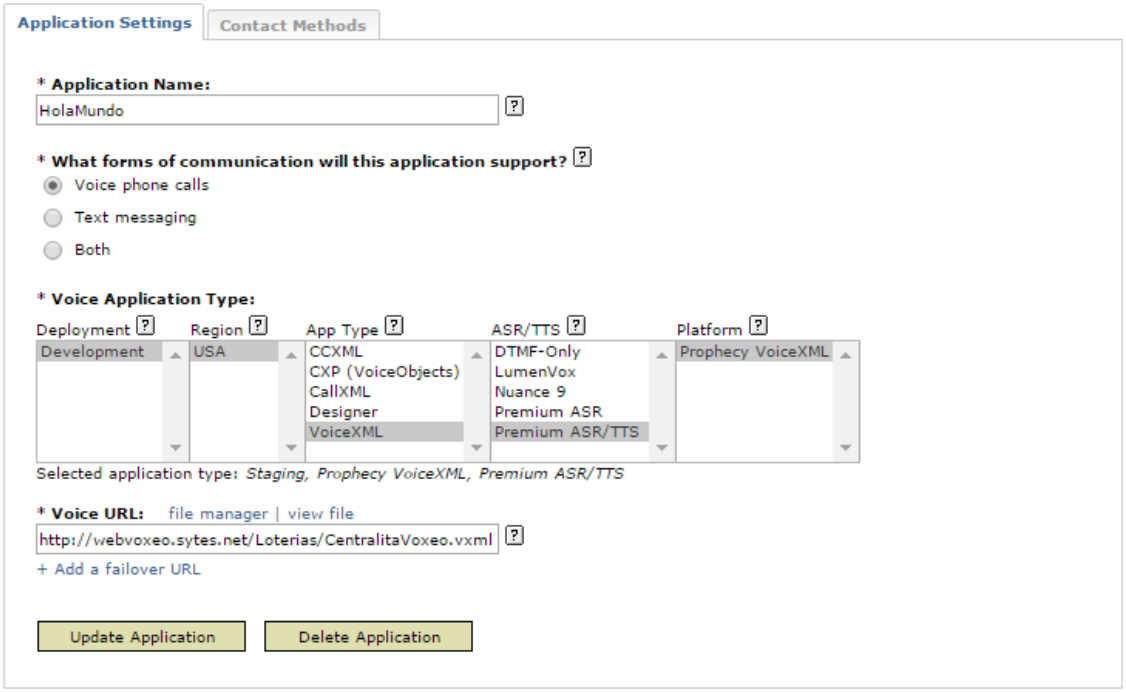

FIGURA 2.35 RESUMEN DE LAS OPCIONES DE NUESTRA APLICACIÓN

Además, existe una pestaña llamada *Contact Methods*, que indica las diferentes formas y tecnologías con las que se puede llamar a la aplicación para realizar las pruebas pertinentes.

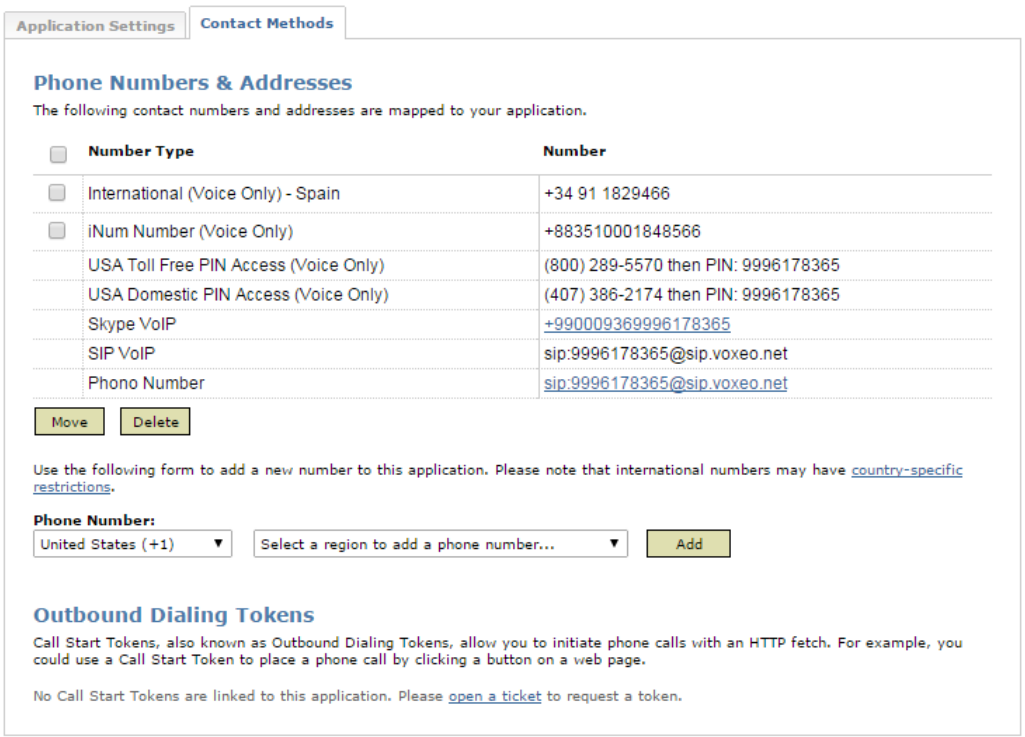

FIGURA 2.36 METODOS DE CONTACTO

Como se puede ver, se puede acceder a la aplicación de diferentes formas. En este proyecto se han utilizado dos de ellas, *Skype VoIP* y *Phono Number*. Para la primera solo se necesita tener instalada en la máquina la aplicación de Skype y para la segunda con cualquier dispositivo móvil o fijo llamando al teléfono indicado se tiene accesibilidad a la aplicación

Capítulo 3

3. Descripción general del sistema

En este capítulo se explican de forma detallada cada uno de los módulos pertenecientes al sistema, así como su arquitectura y funcionalidad. También se explican las diferentes tecnologías utilizadas en el mismo y su pertinente configuración para su correcto funcionamiento.

3.1. Introducción al sistema

El presente proyecto fin de carrera tiene como objetivo hacer más fácil el acceso a la información de las loterías y apuestas del estado. Para conseguirlo, ha sido basado en diferentes tecnologías y lenguajes de desarrollo. Los lenguajes de programación utilizados han sido PHP y el estándar de VoiceXML.

Una descripción general de la interfaz del sistema, sería la siguiente.

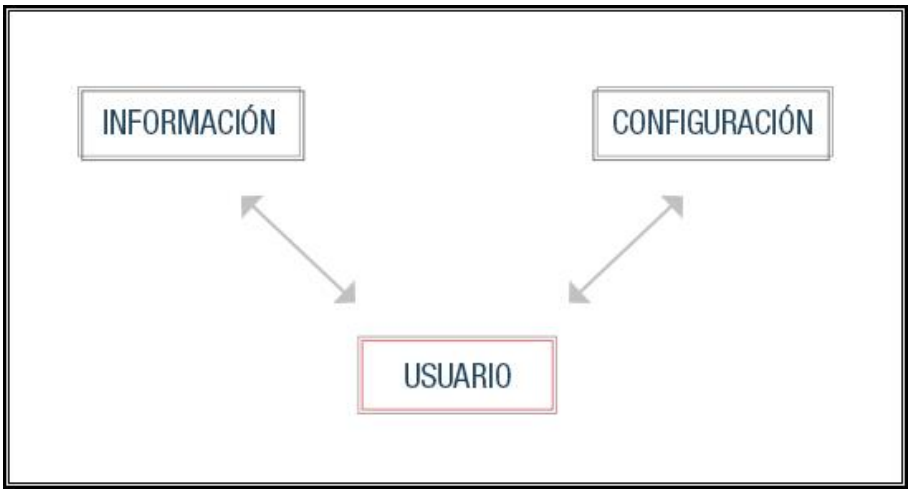

FIGURA 3.1 INTERFAZ DEL SISTEMA

Tal y como se puede ver en la figura anterior, el usuario se comunica con el sistema con dos propósitos diferentes: realizar la configuración del sistema o pedir la información necesaria en ese mismo momento.

Tal y como observamos en la Figura 3.2, la arquitectura de esta aplicación se puede dividir en tres partes bien diferenciadas:

- Servidor VoiceXML. Es el encargado de recibir todos los tipos de peticiones que quieren recibir la información.

- Servidor Web. Es el servidor donde tenemos alojados los ficheros que el servidor VoiceXML necesita interpretar para saber cuál es su funcionamiento.

- Base de datos. En ella es donde se guardan todos los datos necesarios para el buen funcionamiento del sistema.

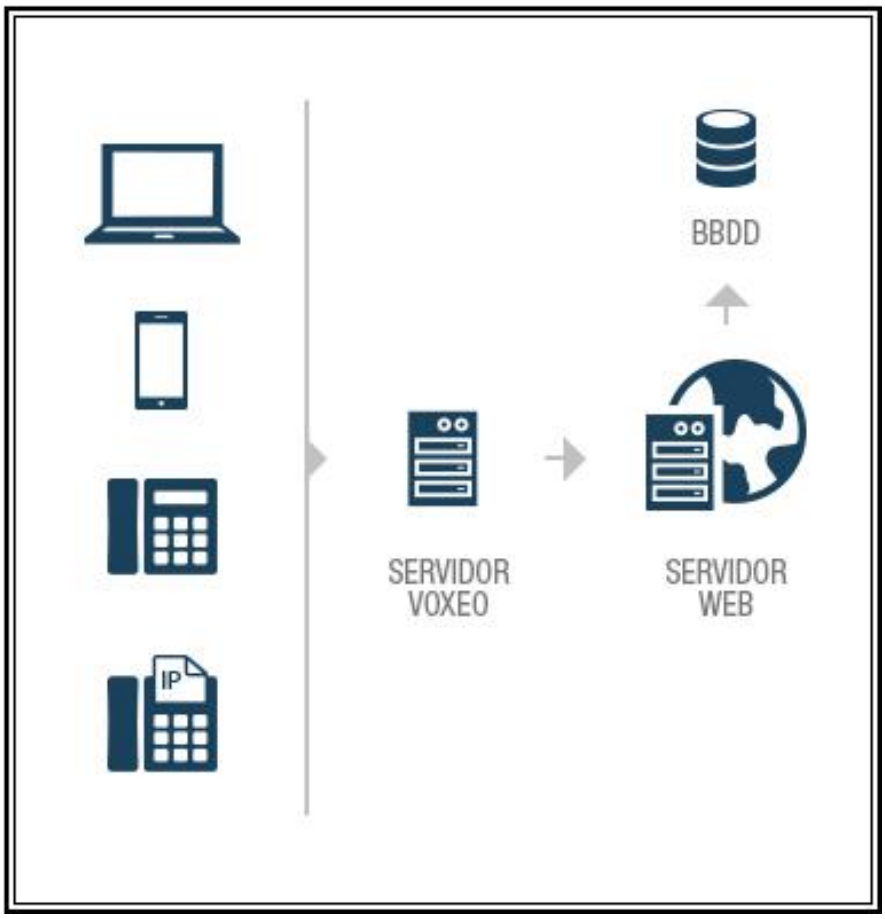

*FIGURA **3.2 DIAGRAMA FLUJO DE DATOS***

El esquema sería muy fácil de interpretar, el usuario en cuestión accede al servidor de Voxeo por el canal del que disponga.

El servidor Voxeo una vez que recibe una petición a su aplicación, accede al servidor Web para saber qué decisión debe de tomar.

Dentro de esta aplicación tenemos diferentes partes que acceden a la base de datos, estos datos ayudan a la aplicación en los diferentes cálculos que debe hacer para devolver la información pedida por el usuario.

Al fin y al cabo se puede definir al servidor de Voxeo únicamente como el intérprete de la aplicación, dado que lo que hace es la interactuación con el usuario, toda la lógica y los datos del sistema están alojados en otro servidor central totalmente diferente.

3.2. Tecnologías

3.2.1. Servidor VoiceXML

Se utiliza el servidor VoiceXML como intérprete de los ficheros de código. Debido a ésto, se han tenido que realizar las acciones pertinentes, explicadas en el Apartado 3.2.1.2 para dar de alta la aplicación en el sistema.

- Dar un nombre a la aplicación.
- Establecer que la aplicación solo va a recibir llamadas telefónicas.
- Configurar las diferentes características del sistema.
- Agregar la dirección URL del servidor central.

Una vez realizados estos pasos, la finalidad de este servidor será únicamente depurar la aplicación en tiempo real y comprobar cuál es la razón de los diferentes fallos provocados durante nuestro desarrollo.

3.2.2. Servidor Xampp

Xampp [XAMPP] es un servidor independiente de la plataforma, que consiste en facilitar la gestión de un servidor web Apache. Xampp también permite gestionar la instalación de los interpretes de lenguajes script: PHP y Perl.

El programa está liberado bajo la licencia GNU y actúa como un servidor web libre capaz de interpretar páginas dinámicas.

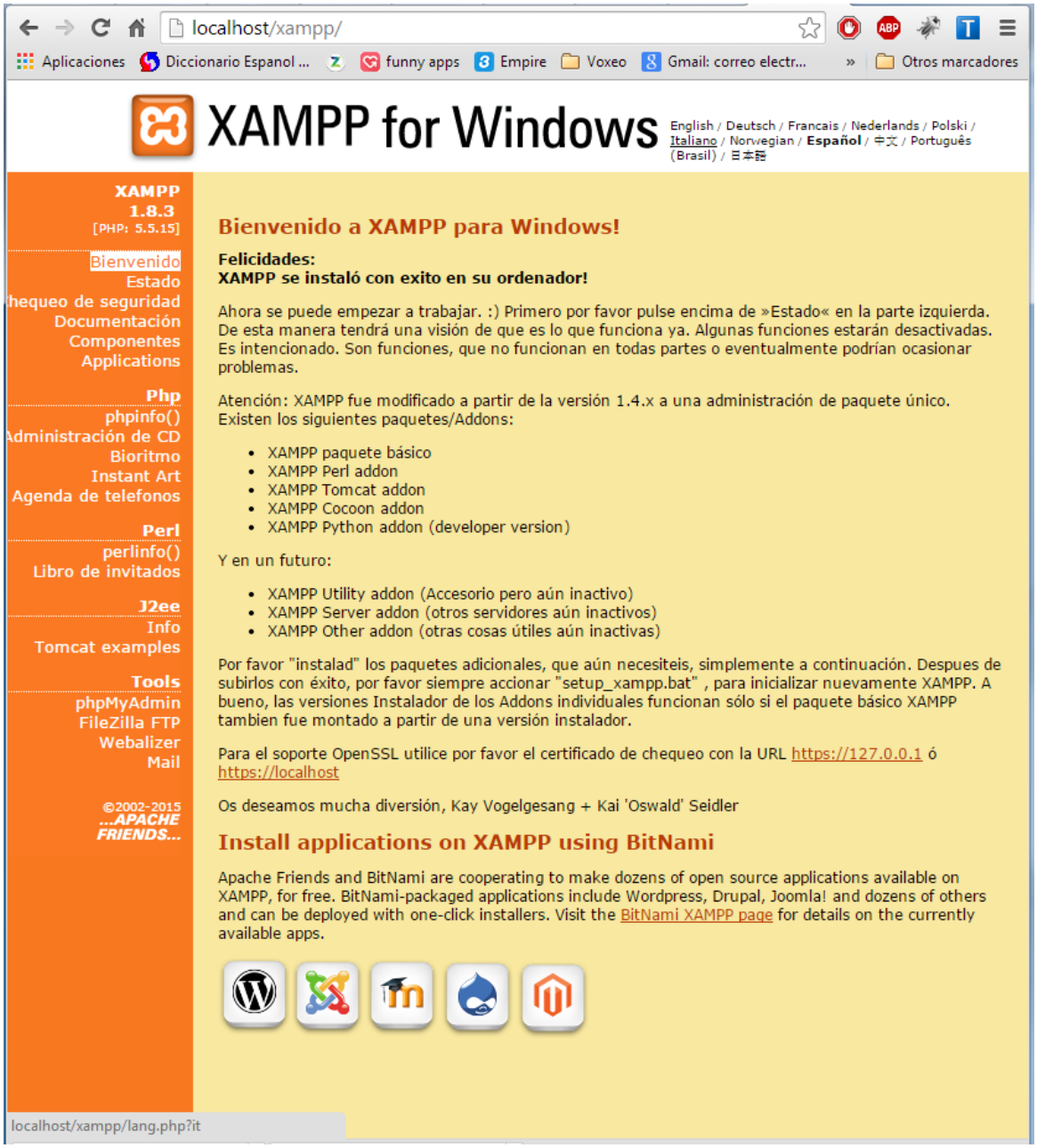

FIGURA 3.3 XAMPP

Esta sería la página utilizada para gestionar todo lo referente al servidor web. Antes de nada se instala esta aplicación en el servidor principal y se le añade el intérprete de la versión de PHP 5.5.15.

Lo principal en este tipo de casos es comprobar, si el servidor que tiene instalado esta aplicación, tiene correcta conectividad con el exterior. Para realizar esto lo único que se tiene que hacer es abrir el puerto 80 del router (normalmente siempre está abierto) y redireccionar todas las peticiones que entren a este puerto a la IP del servidor central.

Se trata de una herramienta sencilla de usar. Solo se deben copiar los ficheros de tu aplicación en la carpeta "unidad_local\xampp\htdocs", y después, si se quiere acceder a ella, se puede hacer desde cualquier navegador y empezar a moverse por las carpetas desde "localhost".

3.2.3. Servidor de Base de datos: MySQL

MySQL es un sistema de base de datos operacional, hoy en día es uno de los más importantes en lo que se refiere al diseño y programación de base de datos de tipo relacional. El programa MySQL se usa como servidor a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo.

Una de las características más interesantes de MySQL es que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a diferentes necesidades y requerimientos. MySQL es capaz de realizar búsquedas de datos e información a alta velocidad.

Como dato histórico, MySQL (My structured Query Language) se remite a principios de la década de 1980. Programadores de IBM lo desarrollaron para contar con un código de programación que permitiera generar múltiples y extendidas bases de datos para empresas y organizaciones de diferente tipo.

Para realizar la gestión de esta base de datos se ha utilizado la herramienta phpMyAdmin. Es un software de código abierto, diseñado para manejar la administración de base de datos a través de una interfaz gráfica de usuario.

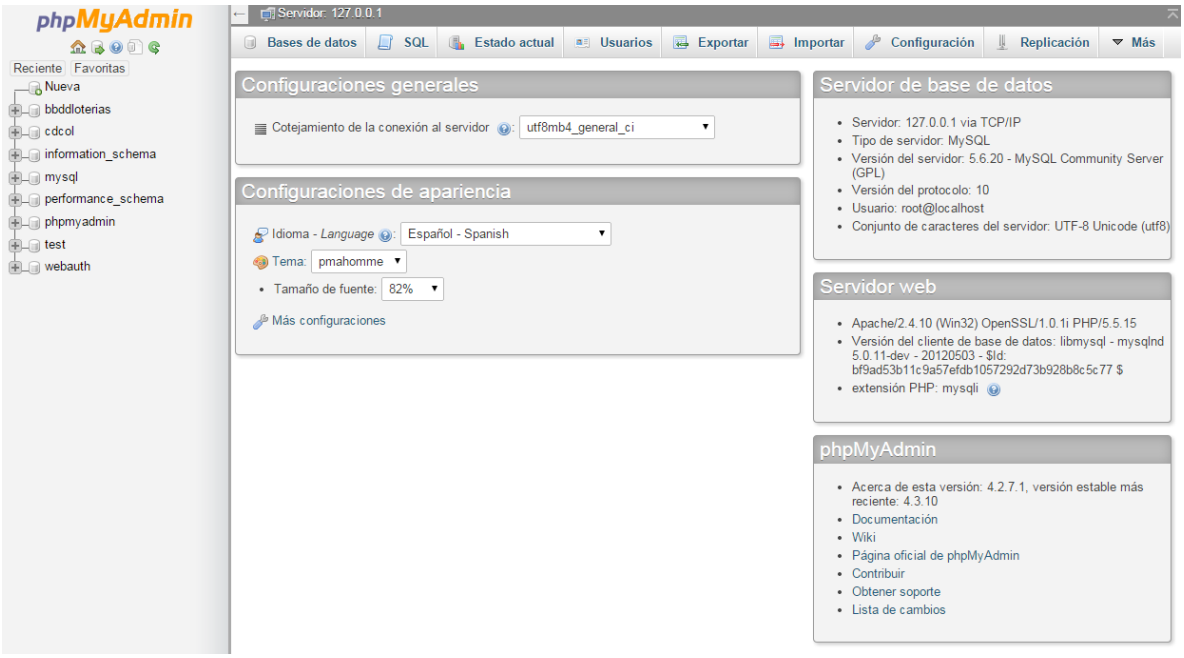

*FIGURA **3.4** **PHPMYADMIN***

La aplicación en si no es más que un conjunto de archivos escritos en PHP que se pueden copiar en un directorio del servidor web, de modo que, cuando se accedan a esos archivos, se muestran unas páginas donde se pueden encontrar las bases de datos y todas sus tablas.

Capítulo 4

4. Descripción detallada del sistema

Este capítulo analiza con detalle cada una de las funcionalidades del sistema. El sistema se divide en módulos, los cuales son explicados definiendo el sentido con el que fueron creados y cómo se ha conseguido la funcionalidad de los mismos.

4.1. Descripción detallada de las base de datos

Tal y como se ha explicado en el tema anterior, la base de datos utilizada es MySQL, gestionada desde la herramienta phpMyAdmin. Esta herramienta permite utilizar su interfaz de usuario para gestionar todo tipo de configuraciones pertenecientes a la base de datos.

De esta manera se procede a detallar cada una de las tablas definidas en el sistema.

FIGURA 4.1 TABLAS DEL SISTEMA

Como puede verse en la Figura 4.1, el sistema depende de tres tablas bien diferenciadas: Loterías, NumerosUsuario, Registro.

- Loterías: Define los diferentes tipos de juegos que están dados de alta en el sistema para informar al usuario sobre sus resultados.

- NumerosUsuario: Tabla utilizada para guardar la información de los sorteros de un usuario en concreto.

- Registro: El sistema guarda un registro de los números de teléfono y el sorteo pedido por el usuario.

Este sería el modelo entidad relación de la base de datos.

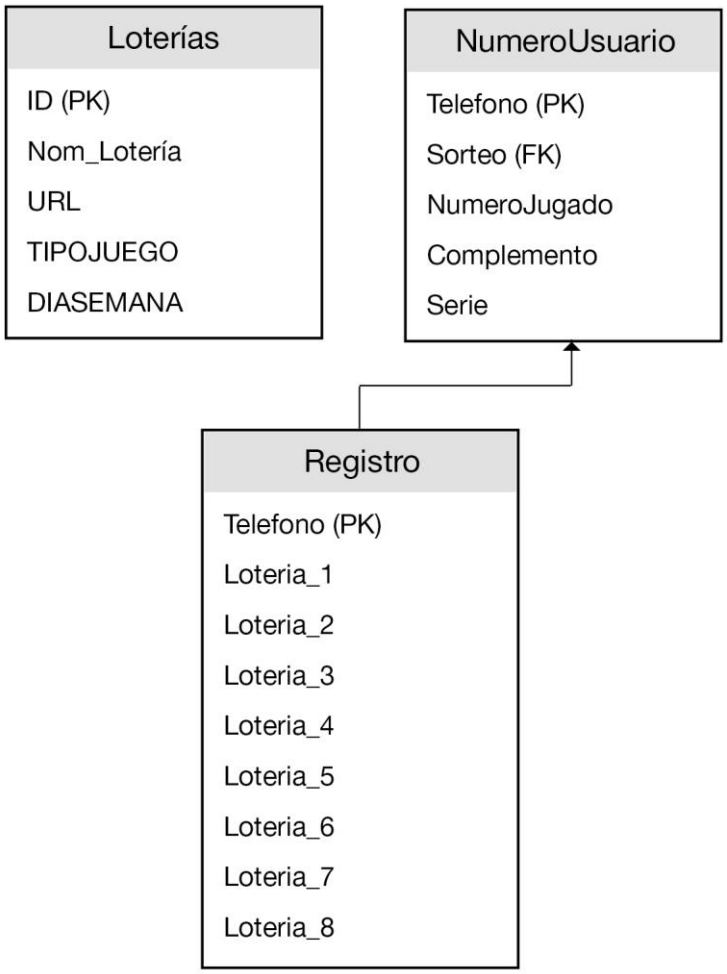

FIGURA 4.2 ENTIDAD RELACIÓN

Todos los accesos a la base de datos del sistema se hacen desde código PHP. VoiceXML no tiene una gestión para realizar accesos a base de datos, por lo que se integra código php en todas las hojas de ejecución para hacer posible esta tarea.

Los accesos a la base de datos son muy sencillos. Se realizan tres tipos de operaciones: INSERT, SELECT y UPDATE. Como su nombre indica, estas operaciones realizan: la operación de inserción a una tabla, la consulta de una tabla particular y el update de una columna en particular.

Para hacer posible la interlocución del sistema con la base de datos, existen diferentes acciones en PHP:

```
<?php

$conexion = mysql_connect($server,$user,$password) or die(mysqli_connect_error());

$selected = mysql_select_db($db,$conexion) or die("Could not select databasename");

?>
```

FIGURA 4.3 CONEXIÓN ***CON LA BASE DE DATOS***

Como puede verse en la Figura 4.3, cualquier conexión con la base de datos se realiza en dos pasos. El primero es establecer la conexión o vínculo con la base de datos, para ello se le pasa como parámetro la IP del servidor BBDD y el usuario con su contraseña. Si ésto no se realiza correctamente, se captura el error y se devuelve al sistema. Si, por el contrario, la comunicación con la base de datos se ha realizado correctamente, se asignará a la conexión la instacia de la base de datos a la que se quiere acceder. El encargado de ello sería la segunda función que aparece en la figura.

El siguiente paso es definir el tipo de operación que se debe ejecutar y mandar la petición a la base de datos. Independedientemente de la operación, todo se realiza de la misma manera, lo único que cambia es la gestión de la variable de respuesta a la hora de realizar una consulta, dado que la BBDD devuelve un ARRAY con todas las columnas coincidientes con la misma.

```
<?php

   $sql = "SELECT id , nom_loteria , URL , TIPOJUEGO FROM LOTERIAS";
$resultados = mysql_query($sql,$conexion);

if($resultados === FALSE)
{
die (mysql_error());
}

if (mysql_num_rows($resultados) != 0)
{
      while($row = mysql_fetch_array($resultados))
      {
            //DATOS DEVUELTOS POR LA BBDD
      }
}

?>
```

*FIGURA 4.4 CONSULTA **A LA TABLA LOTERÍAS***

Como podemos ver en la figura 4.4, se está realizando una consulta a la tabla de LOTERIAS, si el resultado de la consulta devuelve algo, lo recorremos hasta que la variable indique que ya no le quedan más resultados disponibles.

Para terminar, solo debe de cerrar la conexión con la base de datos como se indica en la siguiente figura.

```
<?php

mysql_close($conexion);

?>
```

*FIGURA 4.5 **CIERRE DE LA CONEXIÓN CON LA BASE DE DATOS***

4.2. Descripción detallada del sistema Voice

4.2.1. Introducción al sistema

Tal y como se viene explicando durante el desarrollo del proyecto, el objetivo es realizar una interfaz pública y oral con un sistema de voz que permita devolver al usuario la información requerida sobre los resultados de los sorteos de la lotería.

En la siguiente figura se muestra el esquema general de una interlocución posible entre el usuario y la máquina.

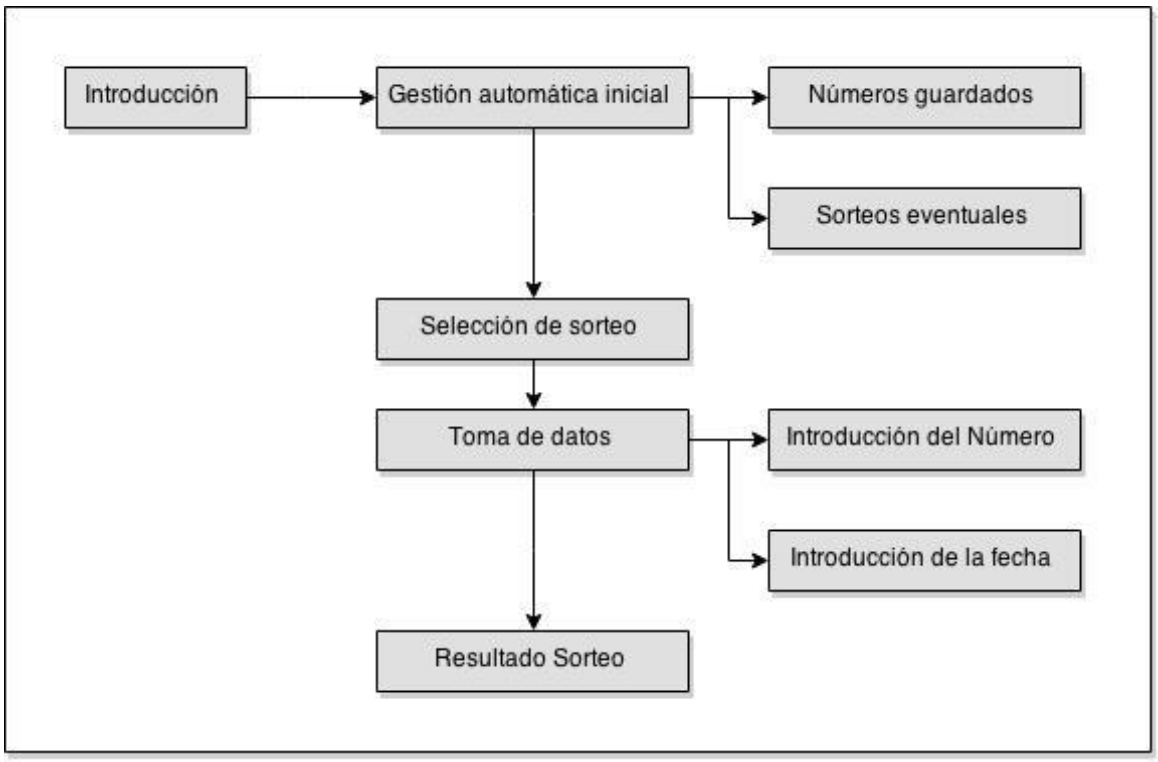

FIGURA 4.6 INTERFAZ MÁQUINA USUARIO

Se puede apreciar que se establecen diferentes puntos de intercambio de información. Muy a alto nivel, el esquema se podría simplificar de la siguiente manera:

El sistema devuelve los resultados de los números que el usuario tenga alojados en él, después devuelve los resultados de los sorteos habituales del usuario conectado. A continuación, pregunta al usuario sobre el sorteo del que desea obtener el resultado. Una vez seleccionado dicho sorteo, el sistema es el encargado de pedir al usuario los datos necesarios para comprobar si el número que juega es el premiado.

Cabe destacar que el usuario puede saltar desde cualquier punto a la gestión inicial automática, además de poder optar a realizar más consultas cada vez que se termine el ciclo. En los siguientes puntos se realizará un estudio detallado de los diferentes módulos y características.

4.2.2. Módulo de presentación

El módulo de presentación es el encargado de dar la bienvenida al usuario al sistema. La principal característica de este módulo es que debe ser capaz de obtener la hora actual del día para calcular si debe decir un saludo u otro (“Buenos días”, “Buenas tardes”, “Buenas noches”).

Para conseguir esta información se vuelve a utilizar el lenguaje PHP. Se hace una llamada a una variable de sistema que devuelve la hora local; una vez obtenida, calculamos la franja horaria en la que el usuario se encuentra y se guarda1 el tipo de saludo en una variable local. Esta variable local es utilizada por el intérprete para realizar el saludo correspondiente.

```
<?php

$horas = getDate();
$saludo;
if ($horas['hours']<14)
 $saludo="Buenos dias.";
if ($horas['hours']>=14)
 $saludo="Buenas tardes.";
if ($horas['hours']>20)
 $saludo="Buenas noches.";

?>
```

*FIGURA 4.7 **FUNCIÓN** PARA **ESTUDIAR LA HORA LOCAL***

Este módulo además se utiliza para obtener el ID del usuario que ha realizado la llamada. Este dato es fundamental en el sistema, permite guardar un registro de quién realiza la llamada para posteriormente realizar los estudios pertinentes.

Este ID puede ser numérico o alfanumérico, dependiendo de la manera a la que se acceda al sistema. Si se está accediendo mediante un teléfono, móvil o fijo, este dato será el número del llamante con el código numérico del país desde el que se realiza la llamada. Si por el contrario la llamada se hace desde cualquier dispositivo VozIP, tal como WhatsApp o Skype, en vez de un número, este ID se representa con el nombre del usuario que está lanzando la llamada.

4.2.3. Módulo de gestión automática inicial

Módulo encargado de realizar el estudio inicial del usuario conectado. El sistema realiza un estudio previo sobre el usuario conectado. Dependiendo de las opciones que previamente tenga guardadas en el sistema un usuario o de las últimas acciones tomadas en el sistema, este módulo se comportará de una manera o de otra.

4.2.3.1. Módulo números guardados

Tal y como su nombre indica, este módulo estudia si el usuario tiene números guardados en el sistema. Si el resultado es afirmativo, el sistema pregunta al usuario si desea obtener los resultados de sus números guardados.

En caso de que la respuesta sea afirmativa, entra en funcionamiento nuestro módulo de obtención de resultados.

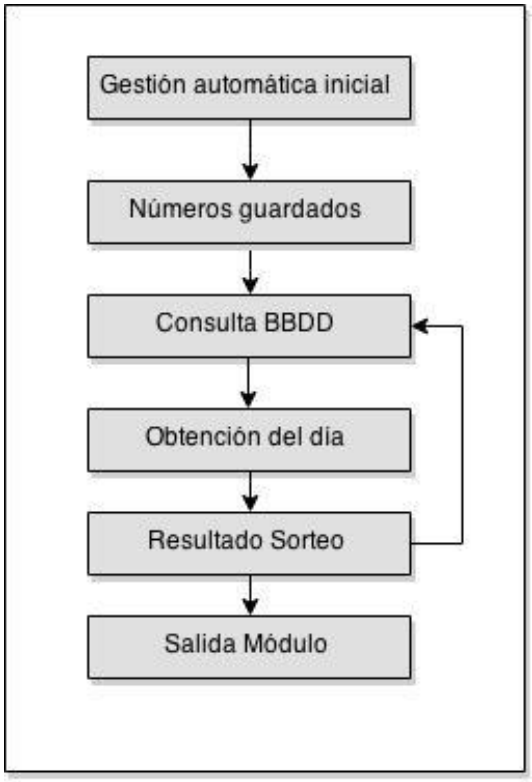

FIGURA 4.8 DIAGRAMA DE SECUENCIA DEL MÓDULO

En este caso, el sistema realizaría una consulta a la tabla NumerosUsuario. Esta consulta puede tener desde ningún resultado (el sistema no preguntaría por la opción de obtener los resultados de los juegos) hasta los que el usuario tenga dados de alta en el sistema. Por lo tanto, el módulo realizaría un bucle con todos los sorteos que el usuario tenga guardados, mostrando en cada uno de ellos si el sorteo ha sido premiado, o si por el contrario, no ha sido ganador del sorteo.

A continuación, en la Figura 4.9 se muestra el flujo de ejecución de las hojas de código php, que hacen que este módulo funcione correctamente.

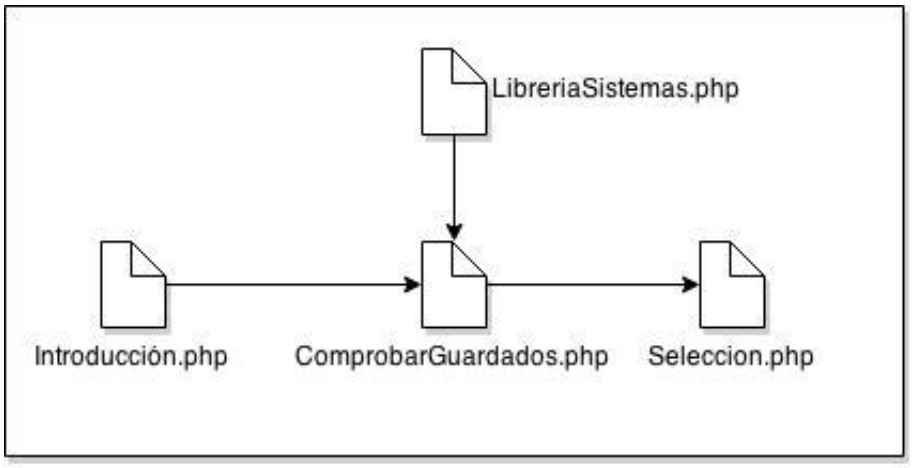

*FIGURA 4.9 **FLUJO DE** EJECUCIÓN **MODULO NÚMEROS GUARDADOS***

Definimos brevemente a qué se dedica cada una de las hojas PHP:

- Introducción.php: Parte del código que obtiene el ID del usuario y estudia si tiene números guardados, si el resultado es positivo, pregunta al usuario si desea saber si sus números guardados son ganadores.

- ComprobarGuardados.php: Esta hoja de código es la que realiza todo el trabajo de este módulo, que es el que se comunica con la base de datos y realiza el estudio de cada uno de los números guardados para saber si ha sido premiado. Este módulo necesita de una librería en la que tenemos definidas una serie de funciones globales para que pueda realizar su función.

- Selección.php: Módulo interlocutor encargado de preguntar al usuario qué desea hacer. En todo el sistema, este módulo se comporta como la página de inicio una vez comenzada la navegación en el mismo.

4.2.3.2. Módulo sorteos habituales

El sistema es capaz de ser proactivo a las llamadas recibidas. Cada vez que un usuario realiza una llamada, este usuario queda registrado en el sistema. Además, cuando este mismo usuario realiza una petición concreta de un juego, el sistema también registra a qué tipo de juego accede.

De esta manera, si en un usuario en concreto existe una predominancia de algún juego en particular, el sistema le pregunta directamente si desea saber el resultado de ese juego.

En este caso, nos estaríamos saltando la hoja de *Selección.php* debido a que el sistema completa ese parámetro por sí solo, sin la necesidad de interactuar con el usuario.

Pero el flujo de toma de datos y de obtención de resultados se ejecuta exactamente igual que el ciclo "normal" del sistema. En el siguiente punto se explican detalladamente estos módulos.

4.2.4. Módulo gestión de sorteos

En este apartado definimos los pasos a seguir por un usuario en el caso de no querer utilizar los módulos automáticos del sistema.

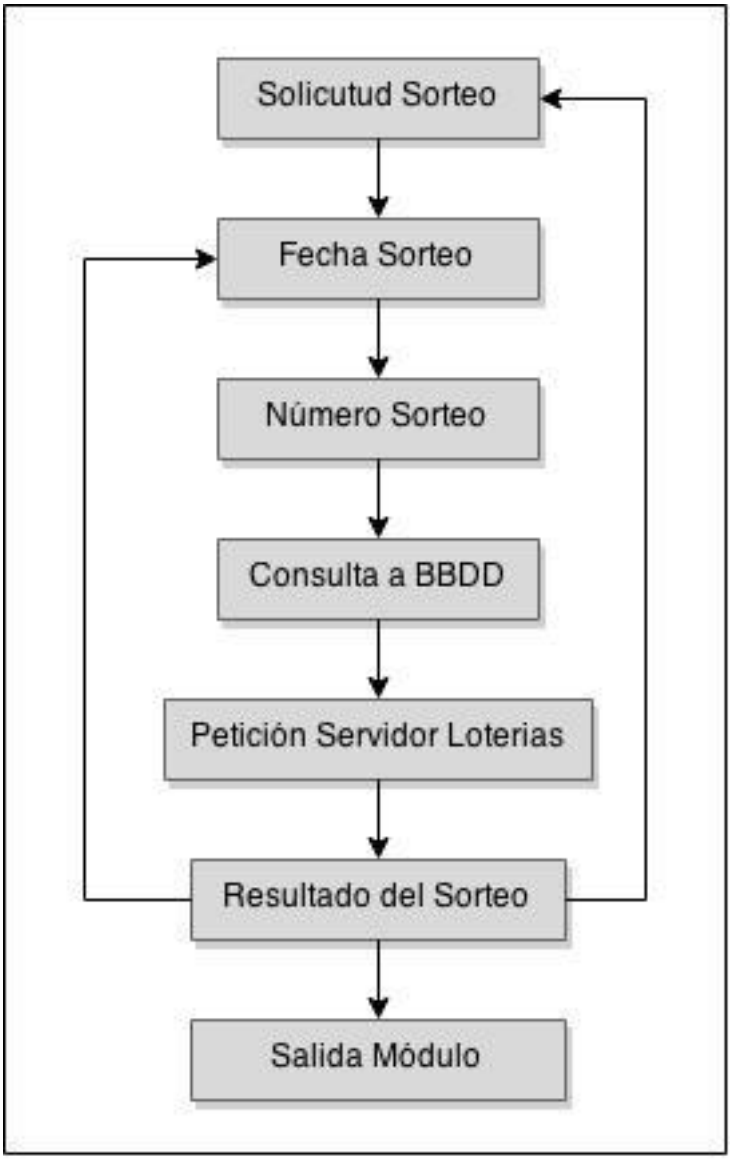

FIGURA 4.10 ***DIAGRAMA DE*** *SECUENCIA,* ***MÓDULO CONSULTA SORTEOS***

4.2.4.1. Módulo de selección de sorteo

A continuación, se explica cómo sería el comportamiento normal del sistema frente a un usuario que no tenga números guardados ni predisposición por un sorteo concreto.

En este apartado toman un valor muy relevante las gramáticas utilizadas, dado que son las encargadas de comprobar que los datos del sistema son correctos.

Cuando un usuario llama al sistema, después de recibir la bienvenida y comprobar la existencia de los dos módulos explicados en los puntos anteriores, el primer paso a realizar es el de pedir al usuario qué es lo que desea hacer: comprobar el número premiado en un sorteo, o guardar un número favorito en el sistema. El sistema, tanto si tiene que guardar un número favorito como si debe responder al usuario por la información de un sorteo, hace lo mismo. En lo único que se diferencia es en la finalidad:

- Si el usuario dice "Guardar": El sistema no devuelve el resultado del sorteo, guarda en el sistema los datos introducidos por el usuario.

- Si el usuario dice "Consultar": El sistema devuelve al usuario el resultado del sorteo seleccionado.

Para realizar esta selección, la aplicación se ayuda de una gramática estática, la cual se ha denominado en el sistema como, *guardaOloteria.grxml*.

En la siguiente imagen se muestra la estructura de la gramática guardaOloteria.grxml. Como se puede observar, la gramática está definida por dos ramas: la rama que define los literales de guardar y la que define los literales de consultar. El funcionamiento es muy sencillo. Es necesario definir en cada una de las ramas todos los literales que se quiera identificar. El sistema comprobará según los literales que diga el usuario a qué rama se está refiriendo, y devolverá un "0" si se quiere guardar o un "1" si lo que se desea es consultar un sorteo.

```
<rule id="miNumero" scope="public">
  <item>
 <one-of>
  <item><ruleref uri="#of_guardar"/><tag><![CDATA[ $ = "0";]]></tag></item>
  <item><ruleref uri="#of_loteria"/><tag><![CDATA[ $ = "1";]]></tag></item>
 </one-of>
  </item>
</rule>

<rule id="of_guardar">
  <item>
    <one-of>
      <item> me gustaria guardar</item>
  <item> guardar </item>
  <item> guardar un numero</item>
  <item> guardar numero</item>
  <item> quiero guardar</item>
  <item> quiero guardar un numero</item>
    </one-of>
  </item>
</rule>

<rule id="of_loteria">
  <item>
    <one-of>
      <item>loteria</item>
  <item>consultar</item>
  <item>quiero consultar la loteria</item>
      <item>quiero consultar un numero de la loteria</item>
      <item>consultar la loteria</item>
      <item>consultar un numero de la loteria</item>
    </one-of>
  </item>
</rule>
```

FIGURA 4.11 GRAMÁTICA, GUARDAOLOTERIA.GRXML

4.2.4.2. Módulo de toma de datos

Una vez que se sabe lo que el usuario desea realizar, pasamos a la toma de los datos del sorteo. Los datos a introducir por el usuario son los siguientes:

- Sorteo a consultar: Nombre del sorteo en el que se desea realizar la acción seleccionada.
- Fecha del sorteo: Este campo sólo es obligatorio en el caso que se desee consultar un sorteo, si se desea guardar, no es preguntado por el sistema.
- Números asociados al sorteo: Engloba todos los números asociados al sorteo definido previamente.

Cada una de estas acciones lleva asociada su gramática estática. En cada una de las peticiones del usuario, se le obliga a completar las peticiones realizadas por el sistema. A continuación se muestra y se explica con detalle cada una de las tres gramáticas utilizadas en este módulo en particular.

En primer lugar se explica la gramática encargada de la selección del juego. Esta gramática está definida en el sistema con el nombre de *juegos.jsgf*.

```
#JSGF V1.0;

grammar juegos;

public <juegos> ="por favor" <peticion> | <peticion> ["por favor"] | <articulo> | <juego> |
salir | ayuda;

<peticion> =  ("me gustaria" | quiero) <verbo>;

<verbo> = (saber | consultar) <articulo>;

<articulo> = (el | la | los) <juego> | (el | la | los) <juego2> | <juego> | <juego2>;

<juego> = <loterias1> | <loterias2> | <loterias3> | <loterias4>;
<juego2> = <loterias5> | <loterias6> | <loterias7> | <loterias8>;

<loterias1>= gordo;
<loterias2>= euromillon | euromillones;
<loterias3>= primitiva;
<loterias4>= "loteria nacional" | nacional;
<loterias5>= bonoloto;
<loterias6>= once;
<loterias7>= lototurf;
<loterias8>= quintuple | "quintuple plus";
```

FIGURA 4.12 GRAMÁTICA, JUEGOS.JSGF

Como se puede ver en esta gramática, se define una estructura inicial para una sentencia de reconocimiento del sistema. Por lo tanto, el sistema es el encargado de reconocer todas y cada una de las combinaciones posibles.

Cabe destacar que todos los literales que están entre paréntesis son aquellos que no son de obligatorio reconocimiento por el sistema. Por lo tanto, si en la entrada del usuario no están, el sistema debe reconocer el juego perfectamente.

La finalidad de esta gramática es reconocer si el usuario introduce como entrada los siguientes sorteos.

- Gordo.
- Euromillón.
- Primitiva.

- Lotería Nacional.
- Bonoloto.
- Once.
- Lototuf.
- Quintuple Plus.

El segundo submódulo de este módulo, es el encargado de reconocer la fecha del sorteo. Esta gramática es mucho más compleja que la anterior, dado que las combinaciones posibles son infinitamente mayores que las de la gramática *juegos.jsgf*. Esta gramática se puede encontrar en el sistema con el nombre de *fecha.grxml*.

En este tipo de gramáticas lo primero que se define es el alcance ("*scope*") de la misma. Como alcance entendemos a todos aquellos elementos que la gramática es capaz de definir. En este caso nuestro alcance son 5 tipos de elementos.

- #day: Se refiere al día seleccionado.
- #mo_of: Las preposiciones que pueden ir delante del mes.
- #mes: En este caso nos referimos al mes seleccionado.
- #of_yr: Las preposiciones que pueden ir delante del año.
- #yr: Se refiere al año seleccionado por el usuario.

Dentro de todas estas reglas, al final, se define el campo <*tag*>. Este campo es el encargado de recoger todos los datos reconocidos por la gramática y devolverlos en el formato definido en el propio campo.

Por lo tanto, el funcionamiento general de la gramática es el siguiente: según el usuario va introduciendo sus datos en el sistema, éste va reconociendo a qué hoja de la gramática se refiere, seleccionando el dato asociado a esta hoja y asignándolo a la variable asociada a la respuesta de la gramática. Una vez que tiene todos los datos recogidos, devuelve un tag con los valores.

```
<tag><![CDATA[ $ = $day + $mes + $yr;]]></tag>
```

Este sería el campo que devolvería la gramática en cuestión. Según va reconociendo el día, mes y año los va agregando a un string en la colocación asignada en el campo tag. Como puede verse en la declaración de la gramática, tanto el campo del día como del mes, están formateados a dos caracteres, es decir, si su valor es un solo número, será formateado con un cero a la izquierda del mismo. De esta manera hacemos posible su reconocimiento posterior en el sistema.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<grammar version="1.0" xml:lang="es-UE" tag-format="semantics/1.0"
        mode="voice" root="mydate">

<rule id="mydate" scope="public">
 <item>
  <item>
   <ruleref uri="#day"/>
   <ruleref uri="#mo_of"/>
   <ruleref uri="#mes"/>
   <ruleref uri="#of_yr"/>
   <ruleref uri="#yr"/>
   <tag><![CDATA[ $ = $day + $mes + $yr;]]></tag>
  </item>
    </item>
</rule>

<rule id="day">
   <item>
  <item><ruleref uri="#numeros1to31" /><tag><![CDATA[ $ = $numeros1to31; ]]> </tag></item>
    </item>
</rule>


<rule id="of_yr">
   <item>
      <one-of>
         <item> en </item>
         <item> en el anio </item>
         <item> de </item>
         <item><ruleref special="NULL"/></item>
      </one-of>
   </item>
</rule>

<rule id="yr">
   <one-of>
      <item> mil novecientos noventa<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "199" + $numeros1to9; ]]></tag></item>
      <item> dos mil<ruleref uri="#numeros1to99"/><tag><![CDATA[ $ = "20" + $numeros1to99;]]></tag></item>
      <item> dos mil<tag><![CDATA[$ = "2000";]]></tag></item>
   </one-of>
</rule>

<rule id="numeros1to31">
   <one-of>
      <item> treinta y uno<tag><![CDATA[$ = "31";]]></tag></item>
  <item> treinta<tag><![CDATA[$ = "30";]]></tag></item>
      <item> vienti<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "2" + $numeros1to9;]]></tag></item>
  <item> viente<tag><![CDATA[$ = "20";]]></tag></item>
  <item><ruleref uri="#numeros10to19"/><tag><![CDATA[$ = $numeros10to19;]]></tag></item>
  <item><ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "0" + $numeros1to9;]]></tag></item>
   </one-of>
</rule>

<rule id="numeros1to99">
   <one-of>
  <item>noventa y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "9" + $numeros1to9;]]></tag></item>
  <item>noventa<tag><![CDATA[$ = "90";]]></tag></item>
  <item>ochenta y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "8" + $numeros1to9;]]></tag></item>
  <item>ochenta<tag><![CDATA[$ = "80";]]></tag></item>
  <item>setenta y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "7" + $numeros1to9;]]></tag></item>
  <item>setenta<tag><![CDATA[$ = "70";]]></tag></item>
  <item>sesenta y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "6" + $numeros1to9;]]></tag></item>
  <item>sesenta<tag><![CDATA[$ = "60";]]></tag></item>
  <item>cincuenta y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "5" + $numeros1to9;]]></tag></item>
  <item>cincuenta<tag><![CDATA[$ = "50";]]></tag></item>
  <item>cuarenta y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "4" + $numeros1to9;]]></tag></item>
  <item>cuarenta<tag><![CDATA[$ = "40";]]></tag></item>
      <item>treinta y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "3" + $numeros1to9;]]></tag></item>
  <item>treinta<tag><![CDATA[$ = "30";]]></tag></item>
      <item>viente y<ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "2" + $numeros1to9;]]></tag></item>
  <item>viente<tag><![CDATA[$ = "20";]]></tag></item>
  <item><ruleref uri="#numeros10to19"/><tag><![CDATA[$ = $numeros10to19;]]></tag></item>
  <item><ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "0" + $numeros1to9;]]></tag></item>
   </one-of>
</rule>

<rule id="numeros10to19">
   <one-of>
  <item>diez<tag><![CDATA[$ = "10";]]></tag></item>
      <item>once<tag><![CDATA[$ = "11";]]></tag></item>
  <item>doce<tag><![CDATA[$ = "12";]]></tag></item>
      <item>trece<tag><![CDATA[$ = "13";]]></tag></item>
      <item>catorce<tag><![CDATA[$ = "14";]]></tag></item>
      <item>quince<tag><![CDATA[$ = "15";]]></tag></item>
      <item>dieciseis<tag><![CDATA[$ = "16";]]></tag></item>
      <item>diecisiete<tag><![CDATA[$ = "17";]]></tag></item>
      <item>dieciocho<tag><![CDATA[$ = "18";]]></tag></item>
      <item>diecinueve<tag><![CDATA[$ = "19";]]></tag></item>
   </one-of>
</rule>

<rule id="numeros1to9">
   <one-of>
      <item> uno<tag><![CDATA[$ = "1";]]></tag></item>
  <item> dos<tag><![CDATA[$ = "2";]]></tag></item>
      <item> tres<tag><![CDATA[$ = "3";]]></tag></item>
      <item> catro<tag><![CDATA[$ = "4";]]></tag></item>
      <item> cinco<tag><![CDATA[$ = "5";]]></tag></item>
      <item> seis<tag><![CDATA[$ = "6";]]></tag></item>
      <item> siete<tag><![CDATA[$ = "7";]]></tag></item>
      <item> ocho<tag><![CDATA[$ = "8";]]></tag></item>
      <item> nueve<tag><![CDATA[$ = "9";]]></tag></item>
   </one-of>
</rule>

<rule id="mes">
   <one-of>
      <item> enero <tag><![CDATA[$="enero";]]></tag></item>
  <item> febrero <tag><![CDATA[$="febrero";]]></tag></item>
      <item> marzo <tag><![CDATA[$="marzo";]]></tag></item>
  <item> abril <tag><![CDATA[$="abril"; ]]></tag></item>
      <item> mayo <tag><![CDATA[$="mayo"; ]]></tag></item>
  <item> junio <tag><![CDATA[$="junio"; ]]></tag></item>
      <item> julio <tag><![CDATA[$="julio"; ]]></tag></item>
      <item> agosto <tag><![CDATA[$="agosto";  ]]></tag></item>
  <item> septiembre <tag><![CDATA[$="septiembre";]]></tag></item>
      <item> octubre <tag><![CDATA[$="octubre";]]></tag></item>
  <item> noviembre <tag><![CDATA[$="noviembre"; ]]></tag></item>
      <item> diciembre <tag><![CDATA[$="diciembre"; ]]></tag></item>
   </one-of>
</rule>

<rule id="mo_of">
 <one-of>
  <item>de</item>
  <item>del mes de</item>
  <item>del mes</item>
  <item><ruleref special="NULL"/></item>
 </one-of>
</rule>
</grammar>
```

FIGURA 4.13 ***GRAMÁTICA, FECHA.GRXML***

El tercer submódulo asociado al sistema es el encargado del reconocimiento de los números relacionados con el sorteo seleccionado. La estructura de esta gramática es muy parecida a la anterior. El único problema que presenta este tipo de gramáticas es la

complejidad que tienen para el intérprete a la hora del reconocimiento de los números. En castellano, los matices sonoros de los números son muy similares, por lo que es complicado para el intérprete reconocerlos si no se presta especial atención a la definición de los literales numéricos.

A continuación mostramos el "*scope"* definido por nuestra gramática *numero.grxml*.

```
<rule id="miNumero" scope="public">
  <item>
 <one-of>
  <item><ruleref uri="#numeros1to99999"/><tag><![CDATA[ $ = $numeros1to99999;]]></tag></item>
  <item><ruleref uri="#miles"/><tag><![CDATA[ $ = $miles;]]></tag></item>
  <item><ruleref uri="#numeros1to999"/><tag><![CDATA[ $ = $numeros1to999;]]></tag></item>
  <item><ruleref uri="#numeros1to99"/><tag><![CDATA[ $ = $numeros1to99;]]></tag></item>
  <item><ruleref uri="#numeros1to9"/><tag><![CDATA[ $ = $numeros1to9;]]></tag></item>
  <item><ruleref uri="#numeros1to99"/><ruleref uri="#numeros1to99_2"/><ruleref uri="#numeros1to99_3"/>
                  <ruleref uri="#numeros1to99_4"/><ruleref uri="#numeros1to99_5"/><ruleref uri="#numeros1to99_6"/><tag><![CDATA[
   $ = $numeros1to99 + " " + $numeros1to99_2 + " " + $numeros1to99_3 + " " + $numeros1to99_4 + " " +
                  $numeros1to99_5 + " " + $numeros1to99_6 ;
  ]]></tag></item>
  <item><ruleref uri="#numeros1to99"/><ruleref uri="#numeros1to99_2"/><ruleref uri="#numeros1to99_3"/>
                  <ruleref uri="#numeros1to99_4"/><ruleref uri="#numeros1to99_5"/><tag><![CDATA[
   $ = $numeros1to99 + " " + $numeros1to99_2 + " " + $numeros1to99_3 + " " + $numeros1to99_4 + " " +
                  $numeros1to99_5 ;
  ]]></tag></item>
 </one-of>
  </item>
</rule>
```

FIGURA 4.14 ***GRAMÁTICA, NUMERO.GRXML***

La principal característica de esta gramática es la cantidad de posibilidades ofrecidas al usuario para el reconocimiento del número.

- Número completo. Se trata del nombre como tal del número pronunciado por el usuario. Es decir, si el usuario quiere saber el resultado del número 2.456, lo define como "*dos mil cuatrocientos cincuenta y seis"*.

- Números solitarios. A diferencia del ejemplo anterior, el usuario define individualmente cada uno de los números que componen la cifra total, por lo que definiría el número 2.456 como "*dos cuatro cinco seis".*

Esta gramática es capaz de reconocer los diferentes interfaces de sorteos disponibles en el sistema. No todos los sorteos usan la misma cantidad de números ni el mismo formato. Por lo tanto esta gramática es capaz de reconocer todos y cada uno de ellos. A continuación se explica los diferentes sorteos utilizados y el formato de sus números.

Tal y como se puede observar en la definición de la gramática, tenemos definidas reglas tanto para los números completos como para los números solitarios.

En el primero de los casos, se establecen reglas que engloban los números por número de cifras. Se establecen 5 tipos de reglas diferentes, desde los números de una cifra, hasta los de cinco.

En el segundo caso solo definimos dos reglas: una para los tipos de sorteo que engloban cinco números y otra para los sorteos que engloban seis números. Cuando se realiza este tipo de reconocimiento, limitamos que cada hueco definido para un número no pueda ser mayor de dos cifras.

```
<rule id="numeros1to99">
  <one-of>
 <item> noventa y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "9" + $numeros1to9;]]></tag></item>
 <item> noventa <tag><![CDATA[$ = "90";]]></tag></item>
 <item> ochenta y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "8" + $numeros1to9;]]></tag></item>
 <item> ochenta <tag><![CDATA[$ = "80";]]></tag></item>
 <item> setenta y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "7" + $numeros1to9;]]></tag></item>
 <item> setenta <tag><![CDATA[$ = "70";]]></tag></item>
 <item> sesenta y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "6" + $numeros1to9;]]></tag></item>
 <item> sesenta <tag><![CDATA[$ = "60";]]></tag></item>
 <item> cincuenta y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "5" + $numeros1to9;]]></tag></item>
 <item> cincuenta <tag><![CDATA[$ = "50";]]></tag></item>
 <item> cuarenta y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "4" + $numeros1to9;]]></tag></item>
 <item> cuarenta <tag><![CDATA[$ = "40";]]></tag></item>
     <item> treinta y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "3" + $numeros1to9;]]></tag></item>
 <item> treinta <tag><![CDATA[$ = "30";]]></tag></item>
     <item> viente y <ruleref uri="#numeros1to9"/><tag><![CDATA[$ = "2" + $numeros1to9;]]></tag></item>
 <item> viente <tag><![CDATA[$ = "20";]]></tag></item>
 <item><ruleref uri="#numeros10to19"/><tag><![CDATA[$ = $numeros10to19;]]></tag></item>
 <item><ruleref uri="#numeros1to9"/><tag><![CDATA[$ = $numeros1to9;]]></tag></item>
  </one-of>
</rule>
<rule id="numeros1to9">
  <one-of>
 <item> cero <tag><![CDATA[$ = "0";]]></tag></item>
     <item> uno <tag><![CDATA[$ = "1";]]></tag></item>
 <item> dos <tag><![CDATA[$ = "2";]]></tag></item>
     <item> tres <tag><![CDATA[$ = "3";]]></tag></item>
     <item> catro <tag><![CDATA[$ = "4";]]></tag></item>
     <item> cinco <tag><![CDATA[$ = "5";]]></tag></item>
     <item> seis <tag><![CDATA[$ = "6";]]></tag></item>
     <item> siete <tag><![CDATA[$ = "7";]]></tag></item>
     <item> ocho <tag><![CDATA[$ = "8";]]></tag></item>
     <item> nueve <tag><![CDATA[$ = "9";]]></tag></item>
  </one-of>
</rule>
```

*FIGURA 4.15 **GRAMÁTICA NUMERO.GRXML, DEFINICIÓN NÚMEROS***

En la figura anterior podemos ver la secuencia para ir definiendo los distintos tipos de números posibles. Se definirán de menos a más, es decir, siempre se empezará por la unidad más pequeña y se continuarán definiendo cifras hasta el límite deseado. Tal y como se puede ver, definimos la entrada vocal al sistema y su valor pertinente. Una vez definido ésto, continuamos con las de dos cifras. En este caso solo debemos escribir la parte significativa vocal que distingue a cada uno de ellos, por ejemplo “cuarenta y” y después definir qué tipo de regla es la que le sigue, en este caso, lo enlazaríamos con las reglas de un solo dígito.

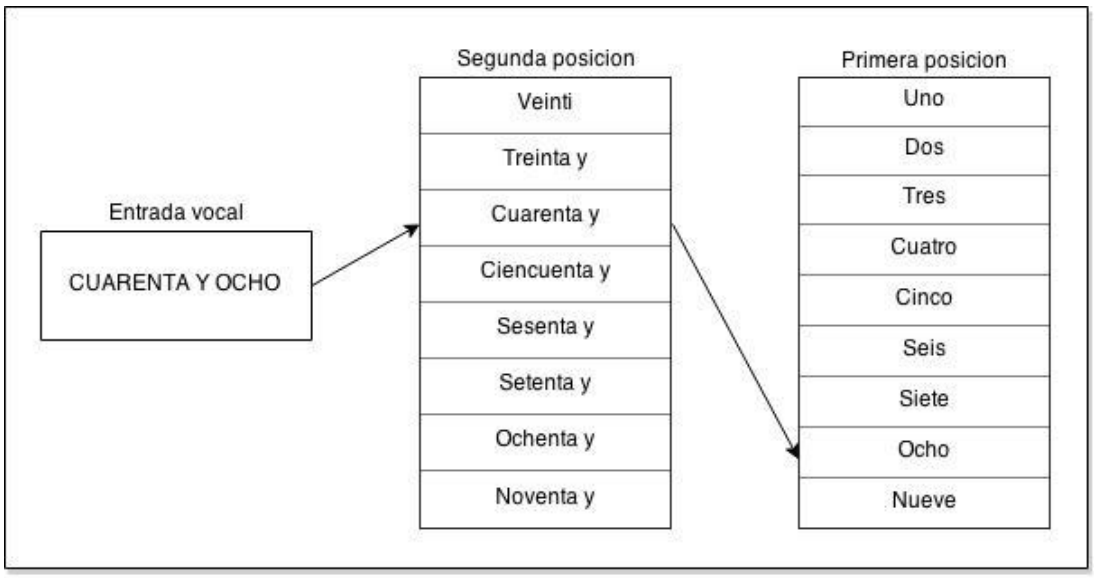

*FIGURA 4.16 **RECONOCIMIENTO NUMÉRICO***

En el ejemplo expresado en la figura 4.16, podemos ver cómo actuaría el reconocimiento de la gramática frente a un número de dos dígitos. La gramática estudiaría de izquierda a derecha la entrada vocal, buscando coincidencias exactas con las definidas en cada uno de los campos, es este caso reconoce que es el "cuarenta y", según la regla definida, pero su trabajo no termina en este momento, debe seguir buscando en la regla enlazada el resto del string introducido por el usuario en el grupo de los números de un solo dígito, que en este caso sería el 8. Como resultado del estudio, el sistema devolvería el valor numérico 48.

4.2.4.3. Módulo comprobar resultados

Una vez realizada toda la toma de los datos, queda por definir la parte más importante del sistema. La obtención de los resultados de la lotería y comprobar si el número introducido por el usuario está premiado o no.

Para realizar este estudio se utilizan una serie de funciones php que ayudan a realizar las gestiones pertinentes para llegar al fin deseado.

Se han desarrollado diferentes hojas de código dependiendo del tipo de sorteo y su parecido con cualquier otro. Las hojas definidas son las siguientes:

- Once.php: Encargada de los resultados de la lotería ONCE.
- Quintuple.php: Encargada de los resultados del juego Quintuple.
- Nacional.php: Encargada de los resultados de la lotería Nacional.

- General.php: Se engloban diferentes sorteos similares entre sí en cuanto a la forma de jugar. Son la bonoloto, el euromillón, la primitiva, el gordo y el lototurf.

El funcionamiento global en todas las hojas de código es el mismo, lo único que cambia es el patrón de búsqueda de los sorteos.

Para poder realizar la búsqueda de los resultados, es necesario tener una página web que nos permita realizar búsquedas de los sorteos por días, y cuyos resultados sean totalmente legibles para realizar las búsquedas pertinentes.

En la base de datos está guardado el patrón URL utilizado para realizar la consulta a las dos páginas Web utilizadas por el sistema: http://www.loteriasyapuestas.es/ y http://once.combinacionganadora.com/. Lo único que hace el sistema es completar la dirección URL con los datos necesarios para que la página web te devuelva los resultados de los días pedidos. En este momento cabe destacar una de las funciones PHP que hace posible el manejo de los datos.

String file_get_contents ( );

Esta función es capaz de obtener toda una página web completa e incrustarla en un string, de esta manera el sistema necesita recorrer este string buscando algún patrón repetitivo que nos indique que estamos ante el número ganador. En el caso de la lotería de la ONCE el patrón que buscamos es el siguiente: "*ctrlNumbers*". El sistema conoce que todo lo que viene después de este literal, son los números premiados del sorteo.

Una vez que el sistema tiene los números premiados, los ordena de menor a mayor. De la misma forma, repite la acción con los números introducidos por el usuario. Una vez completada esta tarea, compara los números entre sí y responde al usuario indicando si su número ha sido premiado o no.

Por último, el sistema pregunta al usuario si desea saber el resultado de otro número o volver al menú principal para consultar algún otro sorteo diferente.

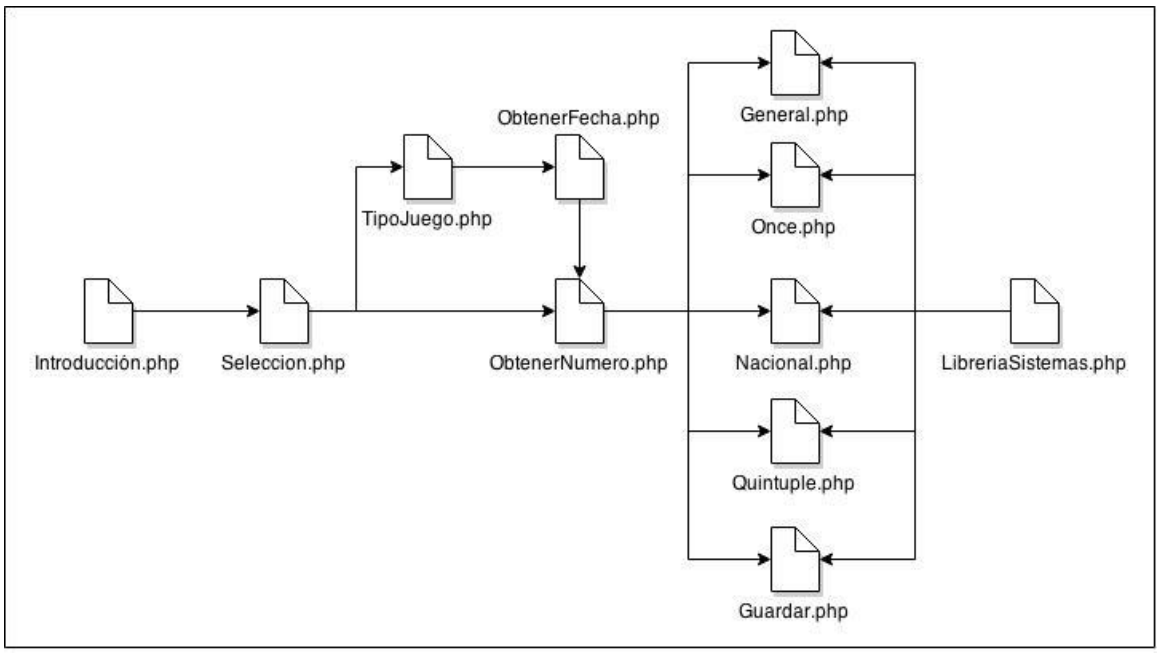

***FIGURA 4.17** FLUJO DE EJECUCIÓN*

En la Figura 4.17 se muestra el flujo de ejecución del módulo completo. Como se puede observar en la figura, desde selección.php se puede pasar a dos ficheros de código diferente. Ésto es debido a que el usuario puede seleccionar si desea guardar un sorteo o consultarlo.

En el primer caso, el flujo iría por *ObtenerNumero.php* cuya secuencia lógica sería ir a *Guardar.php*. Si por el contrario el usuario decide consultar un resultado, tendríamos que tomar la rama que registra esta consulta, *TipoJuego.php* y después pide los datos del sorteo en cuestión.

También en este módulo se utiliza la librería de sistemas para ayudar a realizar la búsqueda de los resultados de las loterías.

Capítulo 5

5. Evaluación del sistema

En este capítulo se describe la evaluación que se ha llevado a cabo para realizar un estudio y valoración subjetiva de la aplicación desarrollada para este proyecto fin de carrera. Se han diseñado una serie de preguntas a 20 usuarios del sistema para que contesten de manera anónima a las mismas. Los resultados han sido analizados para realizar una evaluación del proyecto.

5.1. Evaluación

En este apartado, se explica con detalle el alcance de las preguntas realizadas en la evaluación del proyecto fin de carrera presentado. Con ello, lo único que se desea conseguir es saber si el proyecto en cuestión cumple con los requisitos propuestos inicialmente y si además aporta la funcionalidad para la que ha sido creado.

Se establecieron diferentes líneas de estudio en cada una de las preguntas. Una serie de ellas intentan obtener información sobre si el usuario ya había utilizado algún tipo de sistema parecido al propuesto. Con ello podemos saber si el mismo usuario tiene experiencia en este campo o si por el contrario va a ser una opinión totalmente subjetiva.

Otra línea de estudio es saber si realmente el usuario valora obtener este tipo de información desde diferentes tipos de accesos, como en este caso puede ser el teléfono, tanto móvil como fijo.

Una tercera línea se basa en la calidad del sistema. Es positivo saber si al usuario le ha resultado un sistema amigable, claro y dinámico. Por muy bueno que sea el sistema, no ayuda que los literales que usa el contestador sean incomprensibles o si se hace demasiado lenta la interacción con el mismo. Desperdiciar o gastar demasiado tiempo en

conseguir los resultados penalizaría su utilización y por lo tanto disminuiría la utilidad de este proyecto.

Por último, es bueno saber si en términos generales el usuario está contento con el proyecto propuesto y, sobre todo, si volvería a utilizarlo. Una buena forma para saber si ha sido de su agrado, es preguntar al usuario si realmente recomendaría su utilización a otras personas. Si el usuario está contento con el servicio posiblemente recomiende su utilización.

Por último, se considera interesante preguntar al usuario sobre las posibles mejoras del sistema y así valorar su implementación en el futuro.

Cabe destacar que el formulario se puede encontrar en internet. Para su desarrollo se ha utilizado una herramienta gratuita llamada Survio. Se pueden consultar los resultados creando un identificador en la aplicación. Para poder realizar la encuesta basta con mandar el enlace a las personas que se desee que participen en ella. En nuestro caso, este sería nuestro enlace.

http://www.survio.com/survey/d/M5M9L4W4R2B5K7C0J

A continuación se muestra un ejemplo del formulario usado en el estudio.

Evaluación contestador de Loterías.

Buenos días,

por favor dedique unos minutos de su tiempo para rellenar el siguiente cuestionario.

1 *

¿Ha interactuado alguna vez con un contestador de voz?

- Si
- No

2 *

¿Cómo fué la experiencia general con el sistema?

- Muy buena
- Buena
- Normal
- Mala
- Muy mala

3 *

¿Considera útil la forma de acceso a la información?

- Si
- No
- No sé

4 *

¿Le resultó fácil obtener la información deseada?

- Si
- Más bien si
- Más bien no
- Absolutamente no

FIGURA 5.1 ENCUENTAS, PARTE 1

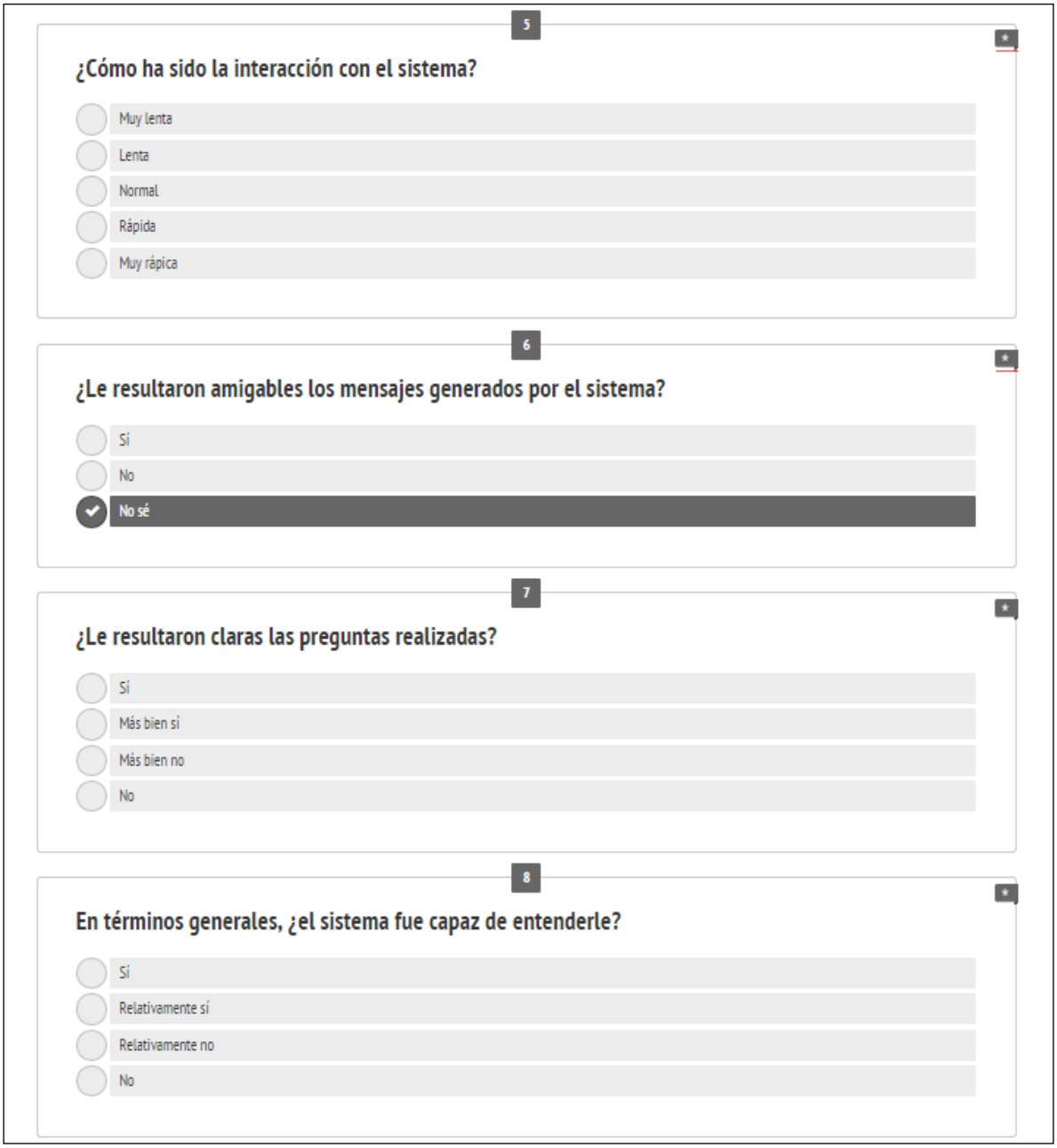

*FIGURA 5.2 **ENCUENTAS, PARTE 2***

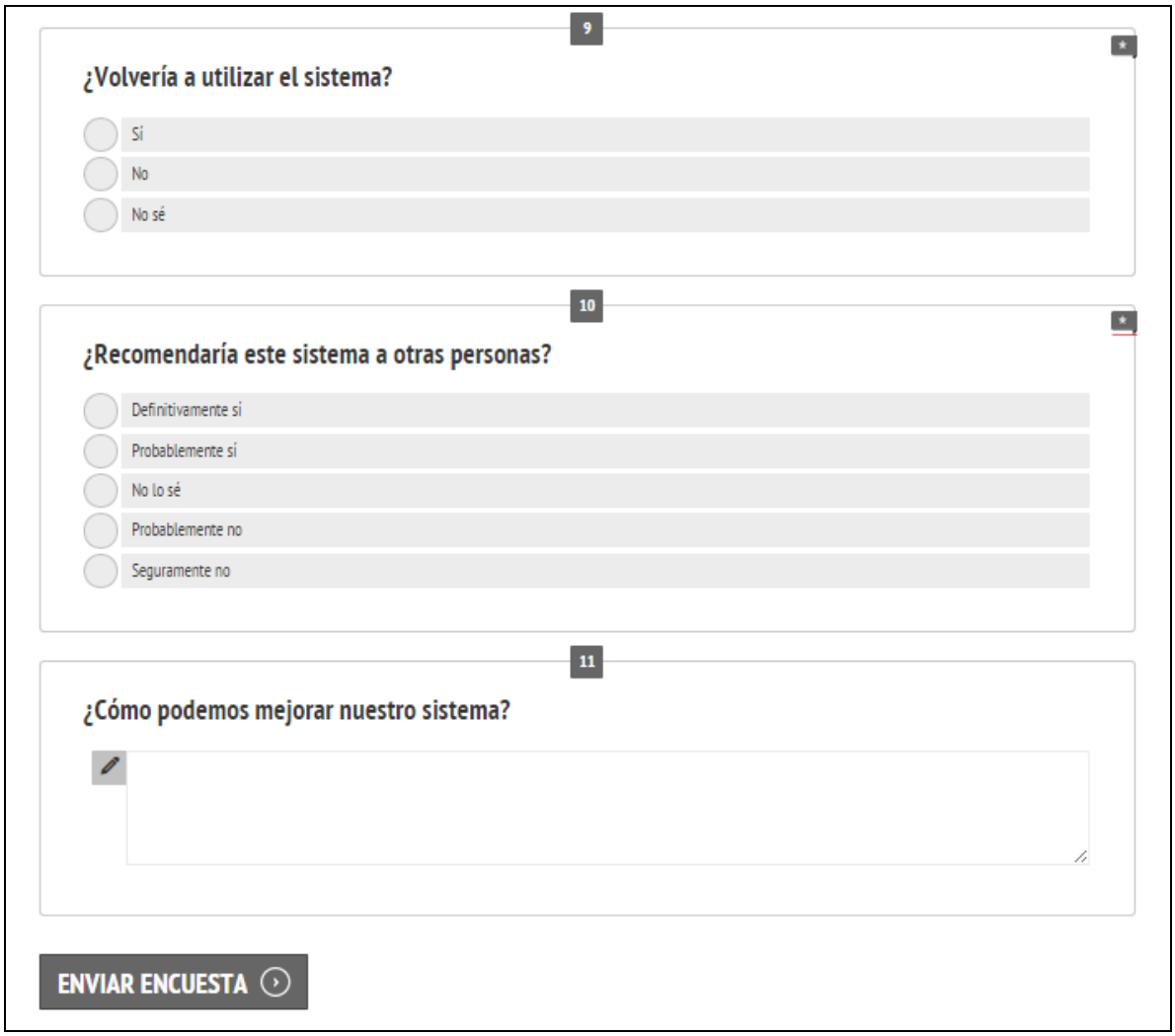

*FIGURA 5.3 **ENCUENTAS, PARTE 3***

5.2. Resultados

La encuesta ha sido realizada a 20 usuarios. Todos ellos sabían la finalidad del proyecto, pero ninguno se hizo a la idea de cómo debía interactuar con el mismo. Cada usuario seleccionó las respuestas que más se adecuaban a lo que pensaba. Algunos de ellos han dejado alguna idea de cómo mejorar el sistema.

A continuación se muestran los resultados de cada una de las preguntas.

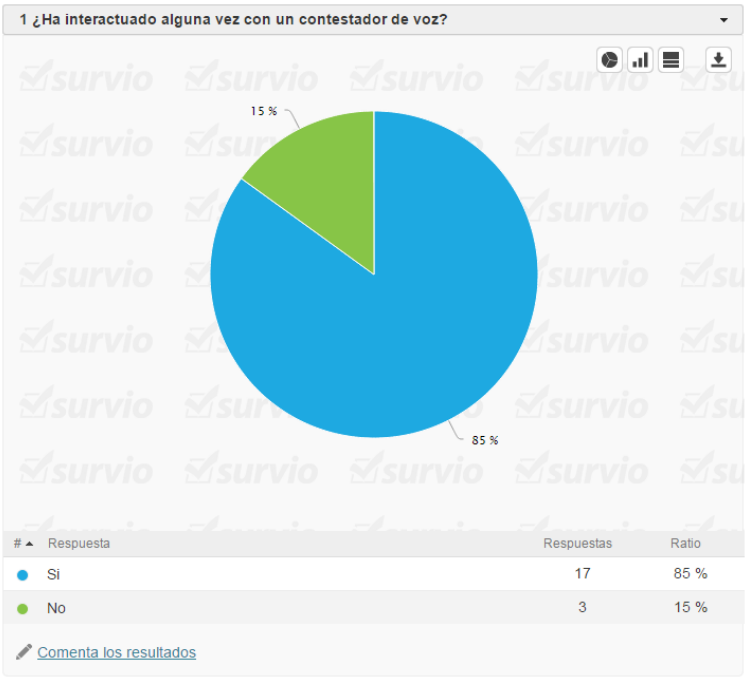

*FIGURA 5.4 **PREGUNTA 1** DE LA ENCUESTA*

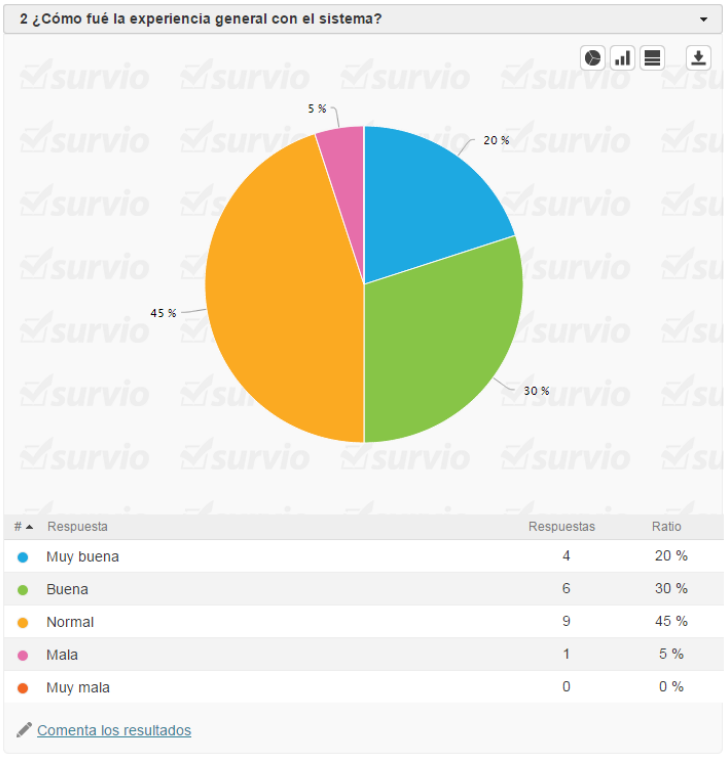

FIGURA 5.5 PREGUNTA 2 DE LA ENCUESTA

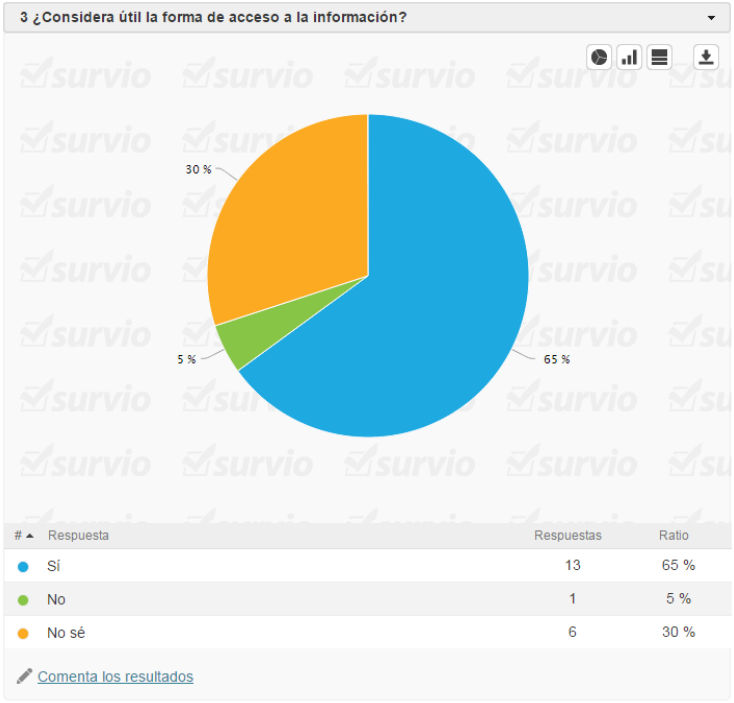

FIGURA 5.6 PREGUNTA 3 DE LA ENCUESTA

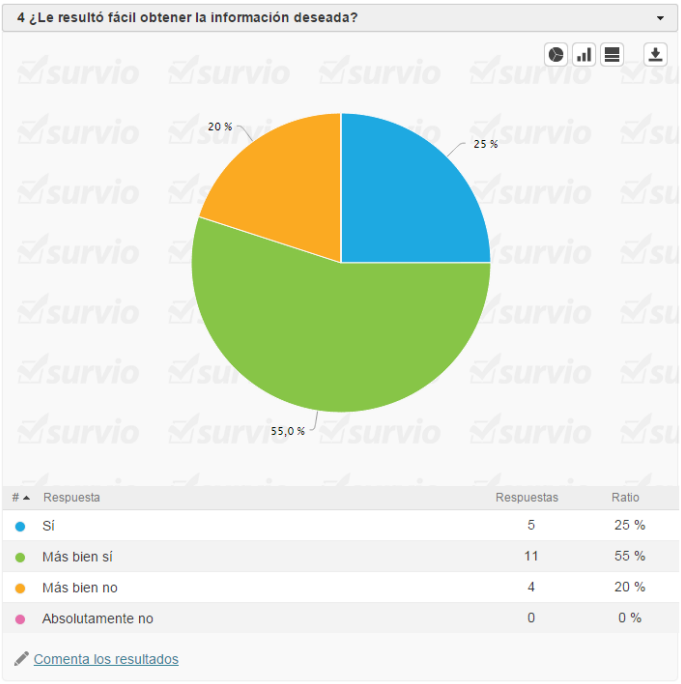

*FIGURA 5.7 PREGUNTA **4** DE LA ENCUESTA*

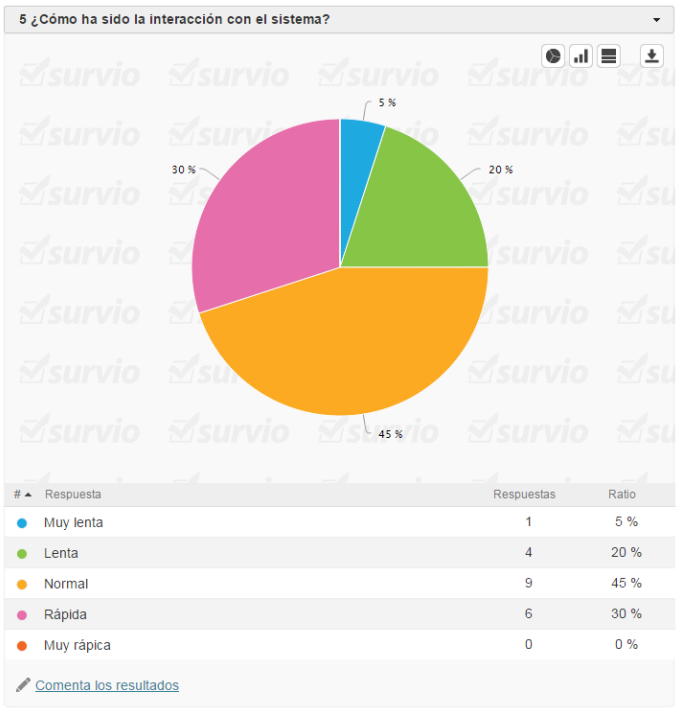

FIGURA 5.8 PREGUNTA 5 DE LA ENCUESTA

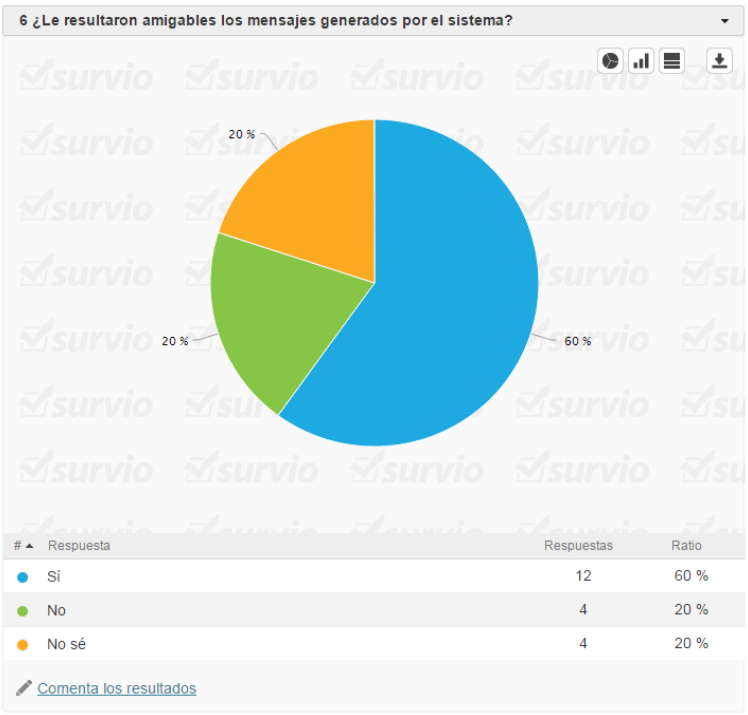

FIGURA 5.9 ***PREGUNTA 6 DE LA ENCUESTA***

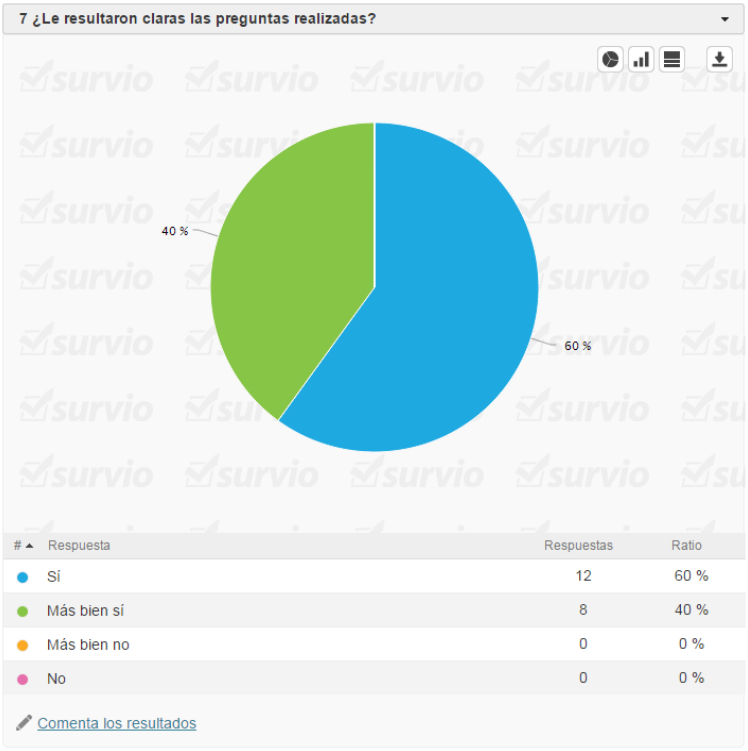

FIGURA 5.10 PREGUNTA ***7 DE LA ENCUESTA***

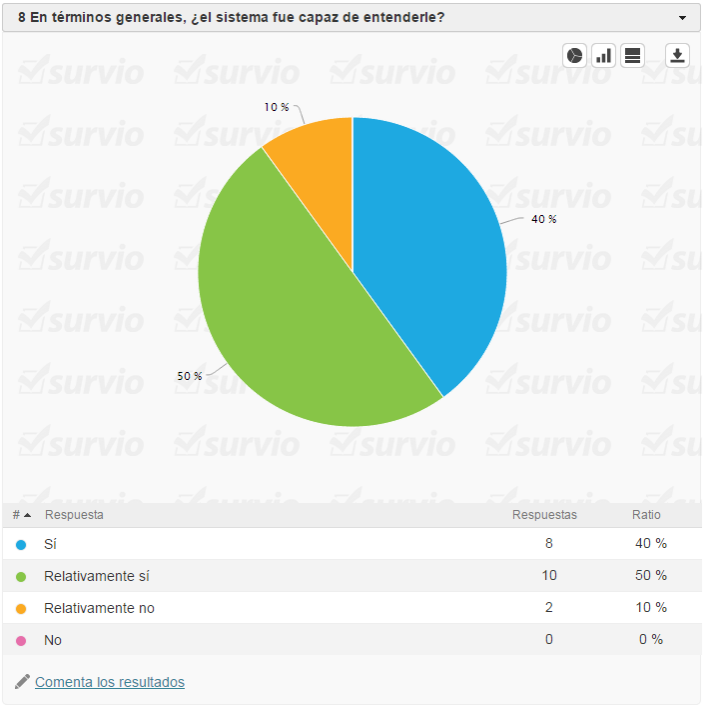

*FIGURA 5.11 PREGUNTA **8** DE LA ENCUESTA*

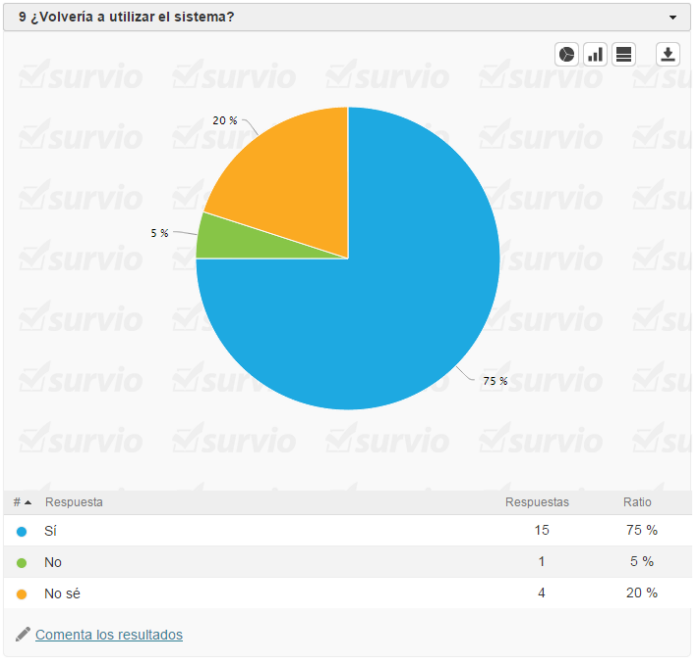

*FIGURA 5.12 PREGUNTA **9** DE LA ENCUESTA*

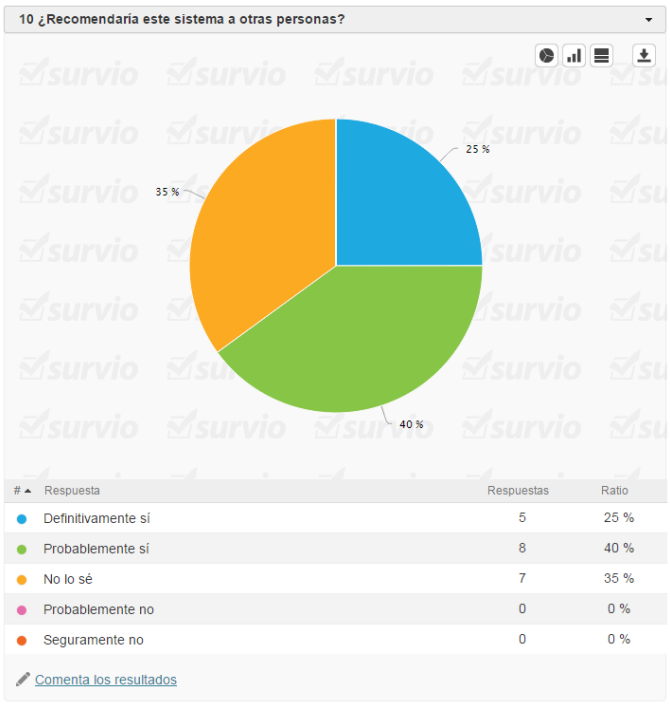

*FIGURA 5.13 PREGUNTA **10** DE LA ENCUESTA*

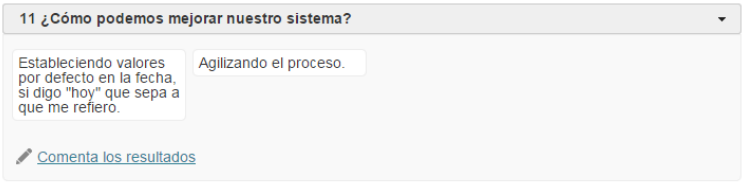

*FIGURA 5.14 **PREGUNTA 11** DE LA ENCUESTA*

Después de estudiar a 20 personas, se pueden sacar bastantes conclusiones sobre el sistema. La primera y nada chocante conclusión, es que la mayoría de los usuarios han interactuado con algún tipo de contestador. Ésto es debido a que actualmente muchas empresas e incluso el estado, los utilizan para realizar la primera toma de contacto con el usuario.

También, en general, la primera toma de contacto con el sistema ha sido muy satisfactoria, ya que la mayoría de los usuarios considera de utilidad esta forma de obtener la información.

En relación al sistema, se han podido sacar diferentes conclusiones, como por ejemplo, que el sistema se expresa con bastante claridad, dado que el 100% ha contestado

que puede entender perfectamente lo que se les pide y sobre todo, que el sistema también les entiende con total claridad, es decir, que el sistema recoge perfectamente los datos introducidos por el usuario.

Como última conclusión podemos decir que, generalmente, todos los usuarios están satisfechos con la labor ofrecida por el sistema, dado que volverían a utilizarlo y además recomendarían su utilización.

Capítulo 6

6. Conclusiones y trabajo futuro

En este capítulo, se explican las conclusiones y posibles mejoras a realizar en el sistema presentado en este proyecto fin de carrera. A partir de los resultados logrados se hace una comparación con los requisitos iniciales propuestos y se puede afirmar que todos ellos han sido cumplidos.

6.1. Conclusiones

Con este proyecto fin de carrera se busca facilitar el acceso a cierta información, ofreciendo un sistema que sea capaz de interactuar con el usuario y comunicarse con él de manera precisa. En este caso, se ofrece la posibilidad de consultar los resultados de la lotería sin tener la necesidad de entrar en una página web o esperar a verlos en la televisión.

En este proyecto se ha planteado la solución con un sistema que fuera capaz de buscar los resultados de los sorteos y mostrarlos al usuario. Además, se ha diseñado un sistema totalmente inteligente capaz de reconocer el número de teléfono de entrada y guardar un registro con los patrones de pedido, para así mostrar directamente los resultados al usuario en la siguiente utilización.

Para llevar a cabo el proyecto, se ha utilizado una base de datos MySQL, la cual cuenta con un diseño preciso que ofrece al sistema una solución más que factible para guardar cada uno de los datos necesarios.

A la hora de realizar la interactuación con el usuario se ha utilizado VoiceXML. Un estándar que nos ofrece una solución tanto a la captación de las entradas vía oral, como a

la generación de elementos sonoros para realizar las comunicaciones oportunas con el usuario. La navegación, dentro de la línea de ejecución, está gestionada por otro lenguaje de programación, PHP. Todo ello acogido bajo la plataforma Voxeo, la cual nos ofrece una serie de herramientas que se han utilizado para el desarrollo y la optimización del sistema.

Como conclusión final, los objetivos establecidos inicialmente han sido alcanzados. Se ha generado un sistema capaz de comunicarse con un usuario y ofrecer la información pedida por el mismo. Con ésto se ha conseguido que algunos usuarios que no están muy familiarizados con las nuevas tecnologías o que por algún motivo no son capaces de utilizarlas, tengan una alternativa real para poder conseguirlo.

6.2. Trabajo futuro

Se muestran los posibles aspectos a trabajar, si se desea mejorar el sistema a corto plazo. Después del estudio de las evaluaciones proporcionadas por el usuario, se han obtenido algunas ideas que se podrían desarrollar.

A la hora de seleccionar la fecha en el sistema, es necesario decir el día, el mes y el año del sorteo. Como mejora, sería posible implementar en la gramática algunas opciones como “hoy”, “ayer”, “antes de ayer”. De esta manera sería mucho más ágil y con menos puntos de error en el sistema.

Otra buena forma de mejorar el sistema sería dotándolo de una forma de pago, de manera que el sistema fuese capaz de comprar los boletos de los usuarios dados de alta.

Valorando esta última mejora, sería necesario tener en cuenta que el sistema no podría almacenar ningún dato de tarjetas, dado que no contemplaría el PCI-DSS y estaría fuera de la ley. Por lo tanto, la solución sería que alguna entidad legal guarde los datos sensibles de las tarjetas de los usuarios y que el sistema sea capaz de realizar las compras oportunas.

Otra mejora, dentro del mismo apartado, sería implementar un sistema de calendario para que el usuario programara cuándo quiere que el sistema compre el boleto y después, establecer unos mismos números a jugar o dejar que el sistema los genere aleatoriamente.

Además de todo lo anterior, se añadiría un sistema de tokenización para vincular a nuestros usuarios con los datos de las tarjetas de crédito de la entidad externa que los guarde. De esta manera, se realiza la gestión y el pago utilizando el token asignado a ese usuario.

Presupuesto

A continuación, se presenta un presupuesto del presente proyecto fin de carrera, teniendo en cuenta tanto el trabajo humano por hora como el material utilizado en el mismo.

Tareas:

Fase 1: Investigación.

- Estudio de los sistemas de diálogo: 7 días.
- Estudio de las tecnologías: 7 días.
- Toma de requisitos: 2 días.
- Viabilidad del sistema: 4 días.

Fase 2: Preparación de entornos.

- Estudio de las herramientas: 4 días.
- Montaje del entorno de desarrollo: 2 días.
- Pruebas unitarias del entorno: 1 día.

Fase 3: Desarrollo.

- Análisis de los requisitos: 7 días.
- Diseño funcional: 7 días.
- Establecimiento de tareas: 4 días.
- Desarrollo de la aplicación: 20 días.
- Pruebas unitarias: 7 días.

- Evaluación de la aplicación: 7 días.

Fase 4: Documentación.

- Memoria del Proyecto Final de Carrera: 30 días.
- Creación de la presentación: 10 días.

Recursos:

Hardware**:**

- Ordenador Sobremesa: 450€
- Micrófono: 10€
- Altavoces: 50€
- Periféricos utilizados: teclado y ratón: 150€

Software:

- Editor de texto Notepad ++: 0 €
- Editor de texto JVoxEdit: 0 €
- Microsoft Office 2010: 218.59€
- Aplicación de comunicación Skype: 0 €
- Servidor independiente de plataforma XAMPP: 0 €
- WBS Editor: 0 €

Dado que el salario establecido para un ingeniero informático es de 35€ la hora, trabajando una jornada laboral completa de 8 horas al día, el presupuesto total de este proyecto es el siguiente:

Total por las tareas:

Descripción	Horas	Coste
Investigación	20 días	5.600 €
Preparación de entornos	7 días	1.960 €
Desarrollo	52 días	14.560 €
Documentación	40 días	11.200 €
Total		33.320€

TABLA 6.1. ELEMENTOS DEL MENÚ

Por lo tanto, el coste del proyecto es el siguiente.

Recurso	Coste
Software	33320 €
Hardware	660 €
Software	218.59€
Subtotal	34198.59€
(21% IVA)	7181.70€
Total	41.380,29€

TABLA 6.2. ELEMENTOS DEL MENÚ

Glosario

DTMF	*Dual-Tone Multi-Frequency*
HTML	*HyperText Markup Language*
IVR	*Interactive Voice Response*
PHP	*Hypertext Pre-processor*
SLS	*Spoken Language Systems*
SQL	*Structured Query Language*
URI	*Uniform Resource Identifier*
VoiceXML	*Voice eXtensible Markup Language*
W3C	*World Wide Web Consortium*
UTF-8 8-bit	*Unicode Transformation Format*
SMF	*Simple Machines Forum*
FTP	*File Transfer Protocol*
API	*Application programming interface*
CCXML	*Call Control Extensible Language*
PCI-DSS	*Payment Application Data Security Standar*

Bibliografia

[LLI06]

Llisterri, J. (2006) "Introducción a los sistemas de diálogo", in LLISTERRI, J.-MACHUCA, M. J. (Eds.) Los sistemas de diálogo. Bellaterra - Soria: Universitat Autònoma de Barcelona, Servei de Publicacions - Fundación Duques de Soria (Manuals de la Universitat Autònoma de Barcelona, Lingüística, 45), págs. 11-21.

[VOICEXML]

“Voice Extensible Markup Language (VoiceXML) Version 2.0”. McGlashan et al. W3C Recommendation 16 March 2004. Disponible:

http://www.w3.org/TR/voicexml20/ [08 de febrero de 2011]

[VOXGATEGAY]

Disponible:

http://www.motorola.com [09 de febrero de 2011]

[VOXPILOT]

Disponible:

http://www.voxpilot.com [09 de febrero de 2011]

[VOXWEBPC]

Disponible:

http://www.voxweb.es/spanish/accesibilidad.html [10 de junio de 2011]

[XHTML+VOICE]

"XHTML+Voice Profile 1.0. ", Jonny Axelsson et al. W3C Recommendation, December 2001. Disponible:

http://www.w3.org/TR/xhtml+voice/ [20 de mayo de 2011]

[XML]

"*Extensible Markup Language (XML) 1.0* ". Bray et al. W3C Recommendation 6 October 2000. Disponible: http://www.w3.org/TR/2000/REC-xml-20001006 [08 de febrero de 2011]

[LAE]

Disponible [INTERNET]: http://www.loteriasyapuestas.es/es

[ONCE]

Disponible [INTERNET]: http://once.combinacionganadora.com

[ZSG+97]

V. Zue, S. Seneff, J.R. Glass, L. Hetherington, E. Hurley, H. Meng, C. Pao, J. Polifroni, R. Schloming, and P. Schmid. From interface to content: Translingual access and delivery of on-line information. In Proceedings of the European Conference on Speech Technology, EuroSpeech, pages 2227–2230, Rhodes, Greece, 1997.

[NEO99]

H. Ney and S. Ortmanns. Dynamic programming search for continuous speech recognition. Signal Processing Magazine, IEEE, 16(5):64–83, Sep 1999.

[War94]

Wayne Ward. Extracting information in spontaneous speech. In Third International Conference on Spoken Language Processing (ICSLP 94), Yokohama, Japan, September 1994.

[FCM+00]

J. Ferreiros, J. Colás, J. Macías-Guarasa, R. de Córdoba, and J.M.Pardo. Control de un equipo de alta fidelidad usando frases habladas de manera natural. In Congreso Iberoamericano IBERDISCAP 2000, pages 187–190, Madrid, Spain, October 2000.

[GTH+05]

D. Griol, F. Torres, L.F. Hurtado, E. Sanchís, and E. Segarra. Different approaches to the dialogue management in the DIHANA project. In 10th International Conference SPEECH and COMPUTER (SPECOM 2005), Patras (GREECE), 2005.

[MCT02]

Michael F. McTear. Spoken dialogue technology: enabling the conversational user interface. ACM Comput. Surv., 34(1):90–169, 2002

[XXH+02]

Weiqun Xu, Bo Xu, Taiyi Huang, and Hairong Xia. Bridging the gap between dialogue management and dialogue models. In Proceedings of the 3rd SIGdial workshop on Discourse and dialogue, pages 201–210, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[GCL+09]

David Griol, Zoraida Callejas, Ramón López, Ana Gutierrez. Utilización de los sistemas de diálogo hablado para el acceso a la información en diferentes dominios. II Conferencia Internacional sobre Brecha Digital e Inclusión Social. 28-30 de octubre de 2009, Leganés, Madrid.

[Gri07]

David Griol. Desarrollo y evaluación de Diferentes Metodologías para la Gestión Automática del Diálogo. Tesis Doctoral. Universidad Politécnica de Valencia, 2007.

BIBLIOGRAFIA

[ARISE]

ARISE (Automatic Railway Information System for Europe) Disponible [INTERNET]: http://cordis.europa.eu/project/rcn/34017_en.html [01-06-1996].

[RAILTEL]

RAITEL (Railway Telephone Information Service) Disponible [INTERNET]: http://cordis.europa.eu/result/rcn/21276_en.html .

[MASK]

MASK (Multimodal-Multimedia Automated Service Kiosk) Disponible [INTERNET]: http://www.limsi.fr/tlp/kiosk-sncf.html [19 de enero de 2013].

[ATIS]

ATIS (Air Travel Information System) Disponible [INTERNET]: http://www.ai.sri.com/natural-language/projects/arpa-sls/atis.html [1990 to 1994].

[MERCURY]

Mercury Disponible [Internet]: http://groups.csail.mit.edu/sls/research/mercury.shtml [19 de enero de 2013].

[PEGASUS]

Disponible [INTERNET]: http://groups.csail.mit.edu/sls/research/pegasus.shtml [04 de febrero de 2011]

[VOYAGER]

Disponible [INTERNET]: http://groups.csail.mit.edu/sls/

[JUPITER]

Disponible [INTERNET]: http://groups.csail.mit.edu/sls/research/jupiter.shtml [04 de febrero de 2011]

[ITSPOKE]

Disponible [INTERNET]: http://oldwww.cs.pitt.edu [04 de febrero de 2011]

[ITSPOKE]

Disponible [INTERNET]: http://acl.ldc.upenn.edu/N/N04/N04-3002.pdf [19 de enero de 2013]

[VOCALIZA]

Disponible [INTERNET]: http://www.vocaliza.es/ [17 de enero de 2015]

[VICO]

Vico (Virtual Intelligent Co-Driver) Disponible [INTERNET]: http://www.cs.cmu.edu/~dgroup/papers/geutner02.pdf [17 de enero de 2015].

[VERBIO]

Disponible [INTERNET]: http://www.verbio.com/webverbiotm/html/demos_cine.php [17 de enero de 2015].

[BBVA]

Disponible [INTERNET]: http://www.naturalvox.com [04 de febrero de 2011].

[SGRS]

"Speech Recognition Grammar Specification Version 1.0 ". Hunt y McGlashan. W3C Proposed Recommendation, December 2003. Disponible: http://www.w3.org/TR/2003/PR-speech-grammar-20031218/ [08 de febrero de 2011].

[URI]

"Uniform Resource Identifiers (URI): Generic Syntax ", IETF RFC 2396, 1998. Disponible: http://www.ietf.org/rfc/rfc2396.txt [08 de febrero de 2011].

[SSML]

"*Speech Synthesis Markup Language Version 1.0* ". Burnett, Walker y Hunt. W3C Candidate Recommendation, December 2003. Disponible: http://www.w3.org/TR/2003/CR-speech-synthesis-20031218/ [08 de febrero de 2011]

[ECMASCRIPT]

"Standard ECMA-262 ECMAScript Language Specification ", Standard ECMA-262, diciembre 1999. Disponible: http://www.ecma-international.org/publications/standards/Ecma-262.htm [08 de febrero de 2011]

[HTTP]

"*Hypertext Transfer Protocol -- HTTP/1.1* ", IETF RFC 2616, 1999. Disponible: http://www.ietf.org/rfc/rfc2616.txt [08 de febrero de 2011]

[XAMPP]

"*Apache MySql Php Perl",* Febrero 2005. Disponible: https://www.apachefriends.org/index.html [08 de febrero de 2011]